US006609153B1

(12) **United States Patent**
**Salkewicz**

(10) **Patent No.: US 6,609,153 B1**
(45) **Date of Patent: Aug. 19, 2003**

(54) **DOMAIN ISOLATION THROUGH VIRTUAL NETWORK MACHINES**

(75) Inventor: **William Salkewicz**, Los Gatos, CA (US)

(73) Assignee: **Redback Networks Inc.**, San Jose, CA (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/220,413**

(22) Filed: **Dec. 24, 1998**

(51) **Int. Cl.**[7] ..... **G06F 15/173**
(52) **U.S. Cl.** ..... **709/223**; 370/352; 370/397
(58) **Field of Search** ..... 709/221, 224, 709/238, 249, 236, 223; 713/201; 370/379, 121, 389, 399, 395, 352, 160, 54, 701, 85, 401, 400, 237, 258, 238; 711/3; 716/8

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | |
|---|---|---|---|---|
| 5,262,906 | A * | 11/1993 | Mazzola | 370/54 |
| 5,309,437 | A * | 5/1994 | Perlman et al. | 370/85 |
| 5,550,816 | A * | 8/1996 | Hardwick et al. | 370/60 |
| 5,604,680 | A * | 2/1997 | Bamji et al. | 716/8 |
| 5,659,796 | A | 8/1997 | Thorson et al. | |
| 5,684,974 | A | 11/1997 | Onodera | |
| 5,692,193 | A | 11/1997 | Jagannathan et al. | |
| 5,737,333 | A * | 4/1998 | Civanlar et al. | 370/352 |
| 5,761,477 | A | 6/1998 | Wahbe et al. | |
| 5,784,707 | A | 7/1998 | Khalidi et al. | |
| 5,832,224 | A * | 11/1998 | Fehskens et al. | 709/223 |
| 5,892,912 | A * | 4/1999 | Suzuki et al. | 709/218 |
| 6,035,105 | A * | 3/2000 | McCloghrie et al. | 709/236 |
| 6,061,349 | A * | 5/2000 | Coile et al. | 370/389 |
| 6,084,892 | A * | 7/2000 | Benash et al. | 370/701 |
| 6,097,719 | A * | 8/2000 | Benash et al. | 370/352 |
| 6,105,027 | A * | 8/2000 | Schneider et al. | 707/9 |
| 6,128,665 | A * | 10/2000 | Iturralde | 709/238 |
| 6,145,011 | A | 11/2000 | Furukawa et al. | |
| 6,167,052 | A * | 12/2000 | McNeill et al. | 370/399 |
| 6,172,981 | B1 * | 1/2001 | Cox et al. | 370/401 |
| 6,219,699 | B1 * | 4/2001 | McCloghrie et al. | 709/221 |
| 6,223,218 | B1 * | 4/2001 | Iijima et al. | 709/221 |
| 6,226,751 | B1 * | 5/2001 | Arrow et al. | 713/201 |
| 6,289,017 | B1 * | 9/2001 | Shani et al. | 370/395 |

OTHER PUBLICATIONS

Adve, V., et al., "Performance Analysis of Mesh Interconnection Networks with Deterministic Routing," IEEE Transactions on Parallel and Distributed Systems, vol. 5, No. 3, pp. 225–246 (3/94).

Almquist, P., "Towards Requirements for IP Routers," Network Working Group, RFC 1716, Downloaded from http://andrew2.andrew.cmu.edu/rfc/rfc1716.html (11/94).

Austin, T., et al., "Efficient Detection of All Pointer and Array Access Errors," Proceedings of the ACM Conference on Programming Language Design and Implementation, pp. 290–301 (6/94).

Bolla, R., et al., "A Neural Strategy for Optimal Multiplexing of Circuit–And–Packet–Switched Traffic," GLOBECOM '92 IEEE Communication Society, pp. 1324–1330 (12/92).

Brooks, R., et al., "An Optimizing Compiler for Lexically Scoped LISP," ACM Symposium on LISP and Functional Programming, pp. 261–275 (8/82).

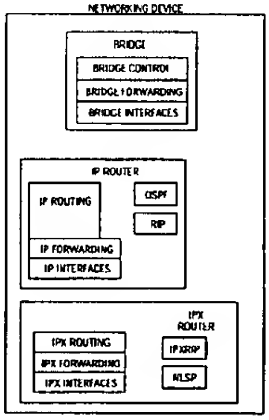
(List continued on next page.)

*Primary Examiner*—Mark R. Powell
*Assistant Examiner*—Thong Vu
(74) *Attorney, Agent, or Firm*—Blakely, Sokoloff, Taylor & Zafman LLP

(57) **ABSTRACT**

A computer implemented method in which Internet Protocol (IP) packets are routed within a first Internet Service Provider's (ISP's) domain from a single network device with a first database. The first database includes addresses of the first ISP. IP packets are also routed within a second ISP's domain from single network device with a second database. The second database, which is separate from the first database, includes addresses of the second ISP.

**81 Claims, 21 Drawing Sheets**



OTHER PUBLICATIONS

Buzen, J.P., et al., "The Evolution of Virtual Machine Architecture," AFIPS National Computer Conference, pp. 291–299 (6/73).

Comer, D., *Computer Networks and Internets,* Prentice Hall (1997).

Cmelik, B., et al., "Shade: A Fast Instruction–Set Simulator for Execution Profiling," ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, pp. 128–137 (5/94).

Dally, W.J., et al., "Deadlock–Free Message Routing in Multiprocessor Interconnection Networks," IEEE Transactions on Computers, vol. C–36, No. 5, pp. 547–553 (5/87).

Dally, W.J., "Virtual–Channel Flow Control," Proc. 17th International Symposium on Computer Architecture, pp. 60–68 (5/90).

Dally, W.J., "Performance Analysis of k–ary n–cube Interconnection Networks," IEEE Transactions on Computers, vol. 39, No. 6, pp. 775–785 (6/90).

Dally, W.J., et al., "Deadlock–Free Adaptive Routing in Multicomputer Networks Using Virtual Channels," IEEE Transactions on Parallel and Distributed Systems, vol. 4 No. 4 (466–475 (4/93).

Davidson, J., et al., "Cint: A RISC Interpreter for the C Programming Language," Proceedings of the SIGPLAN '87 Symposium on Interpreters and Interpretive Techniques, pp. 189–197 (6/87).

Deutsch, L.P., et al., "Efficient Implementation of the Smalltalk–80 System," Proceedings of the 11th Annual ACM Symposium on Principles of Programming Languages, pp. 297–302 (1/84).

Duato, J., "A New Theory of Deadlock–Free Adaptive Routing in Wormhole Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 4, No. 12, pp. 1320–1331 (12/93).

Ellis, J., et al., "Safe, Efficient Garbage Collection for C++," Proceedings of the 1994 USENIX C++ Conference, pp. 143–177 (4/94).

Fischer, C., et al., "The Implementation of Run–Time Diagnostics in Pascal," IEEE Transactions on Software Engineering, vol. SE–6, No. 4, pp. 313–319 (7/80).

Gallager, R., "Scale Factors for Distributed Routing Algorithms," NTC '77 Conference Record (12/77).

Glass, C.J., et al., "The Turn Model for Adaptive Routing," Proc. 19th International Symposium on Computer Architecture, pp. 278–287 (5/92).

Goldberg, R., "Survey of Virtual Machine Research," Honeywell Information Systems and Harvard University, pp. 34–45 (6/74).

Jesshope, C.R., et al., "High Performance Communications in Processor Networks," Proc. 16th International Symposium on Computer Architecture, pp. 150–157 (5/89).

Kirkpatrick, S., et al., "Optimization by Simulated Annealing," Science, vol. 220, No. 4598, pp. 671–680 (5/83).

Leinwand, A., et al., *Cisco Router Configuration,* MacMillan Technical Publishing, Indianapolis, IN (1998).

Li, K., et al., "Memory Coherence in Shared Virtual Memory Systems," ACM Transactions on Computer Systems, vol. 7, No. 4, pp. 321–359 (11/89).

Linder, D.H., et al., "An Adaptive and Fault Tolerant Wormhole Routing Strategy for k–ary n–cubes," IEEE Transactions on Computers, vol. 40, No. 1, pp. 2–12 (1/91).

Luckham, D., et al., "Verification of Array, Record, and Pointer Operations in Pascal," ACM Transactions on Programming Languages and Systems, vol. 1, No. 2, pp. 226–244 (10/79).

May, C., "MIMIC: A Fast System/370 Simulator," Proceedings of the SIGPLAN '87 Symposium on Interpreters and Interpretive Techniques, pp. 1–13 (6/87).

Sites, R., et al., "Binary Translation," Communications of the ACM, vol. 36 No. 2, pp. 69–81, (2/93).

Stallings, W., *High Speed Networks: TCP/IP and ATM Design Principles,* Prentice Hall (1998).

Talia, D., "Message–Routing Systems for Transputer–Based Multicomputers," IEEE Micro, No. 3, pp. 62–72 (6/93).

Wahbe, R., et al., "Efficient Software–Based Fault Isolation," Proceedings of the Symposium 14th ACM on Operating System Principles, 203–216 (12/93).

Yang, C.S., et al., "Performance Evaluation of Multicast Wormhole Routing in 2D–Torus Multicomputers," ICCI '92 IEEE Computer Society, pp. 173–178 (5/92).

Yantchev, J., "Adaptive, Low Latency Deadlock–Free Packet Routing for Networks of Processors," IEEE Proceedings, vol. 136, Pt. E, No. 3, pp. 178–186 (5/89).

* cited by examiner

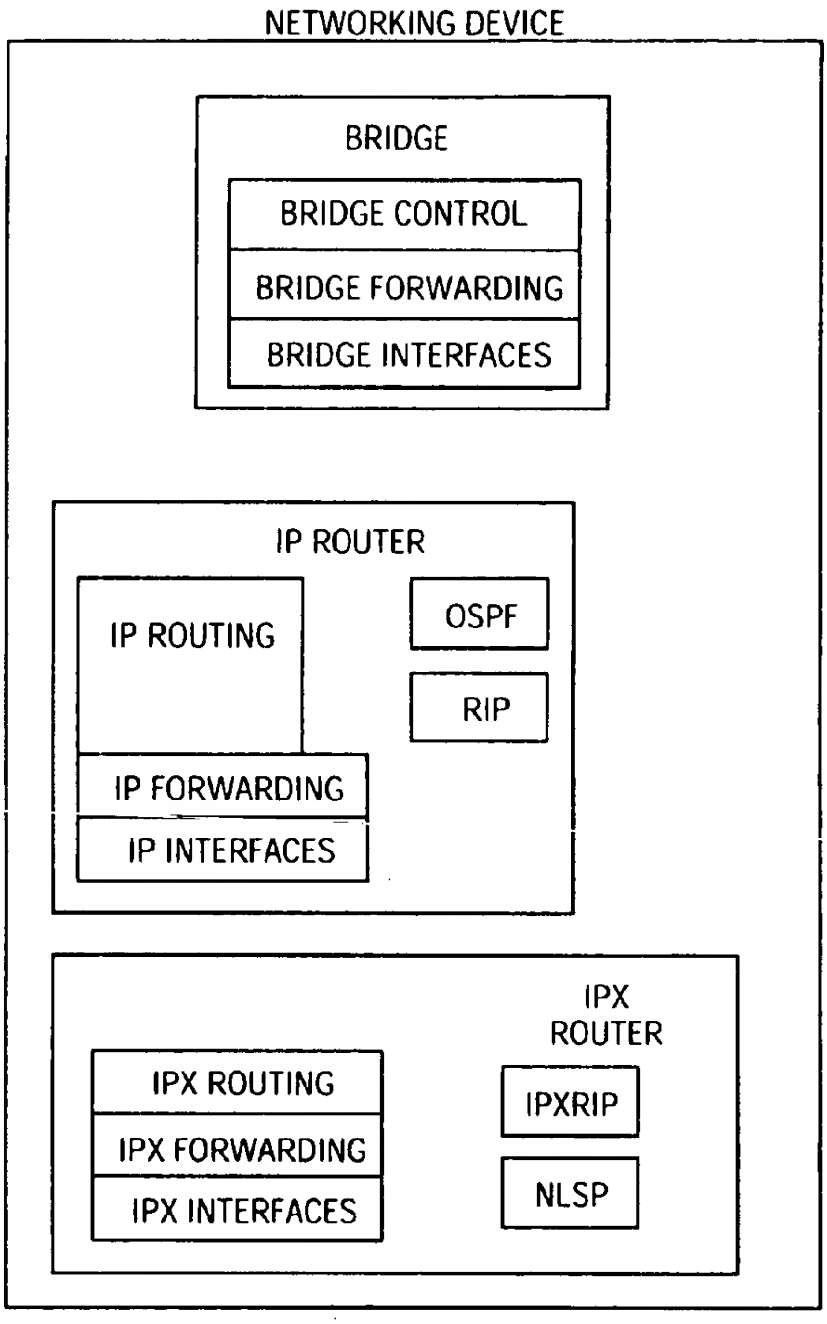

NETWORKING DEVICE
BRIDGE
BRIDGE CONTROL
BRIDGE FORWARDING
BRIDGE INTERFACES
IP ROUTER
IP ROUTING
OSPF
RIP
IP FORWARDING
IP INTERFACES
IPX ROUTER
IPX ROUTING
IPX FORWARDING
IPX INTERFACES
IPXRIP
NLSP

FIG. 1A

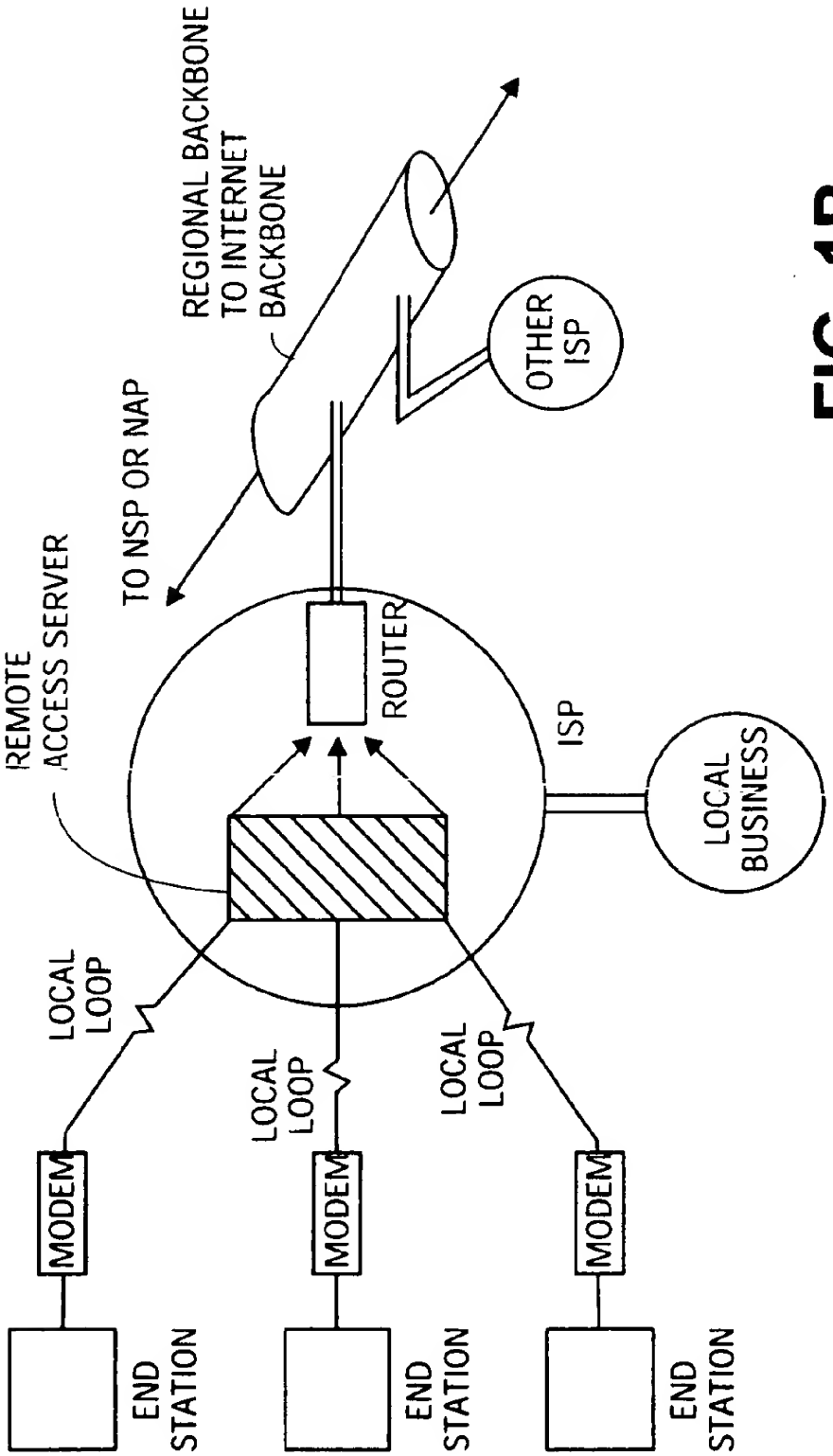
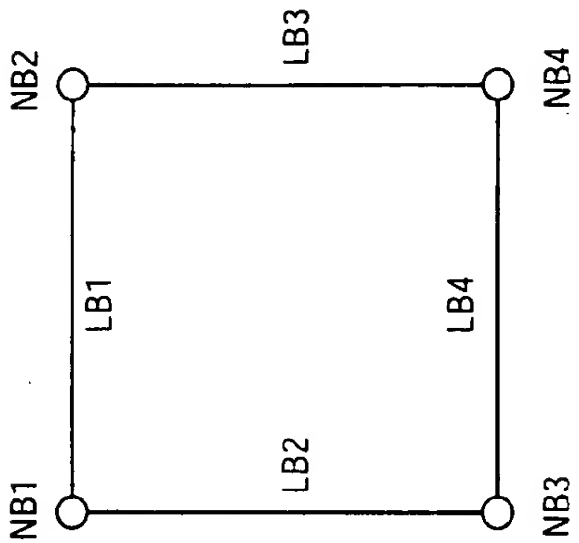

REGIONAL BACKBONE TO INTERNET BACKBONE
OTHER ISP
TO NSP OR NAP
REMOTE ACCESS SERVER
ROUTER
ISP
LOCAL BUSINESS
LOCAL LOOP
LOCAL LOOP
LOCAL LOOP
MODEM
MODEM
MODEM
END STATION
END STATION
END STATION

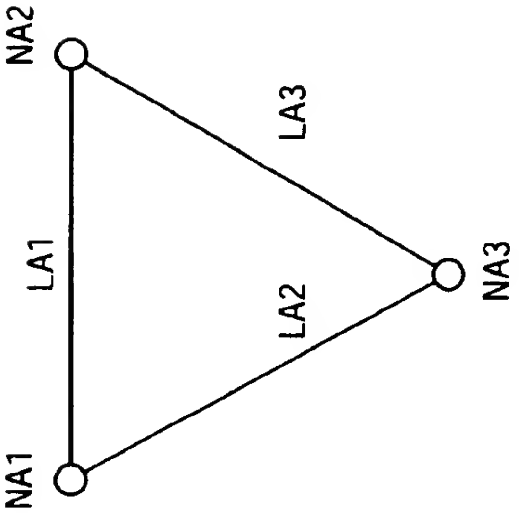
FIG. 1B




NODES NB1, NB2, NB3, NB4

LINKS LB1, LB2, LB3, LB4

FIG. 2B




NETWORK A

NODES NA1, NA2, NA3

LINKS LA1, LA2, LA3

FIG. 2A

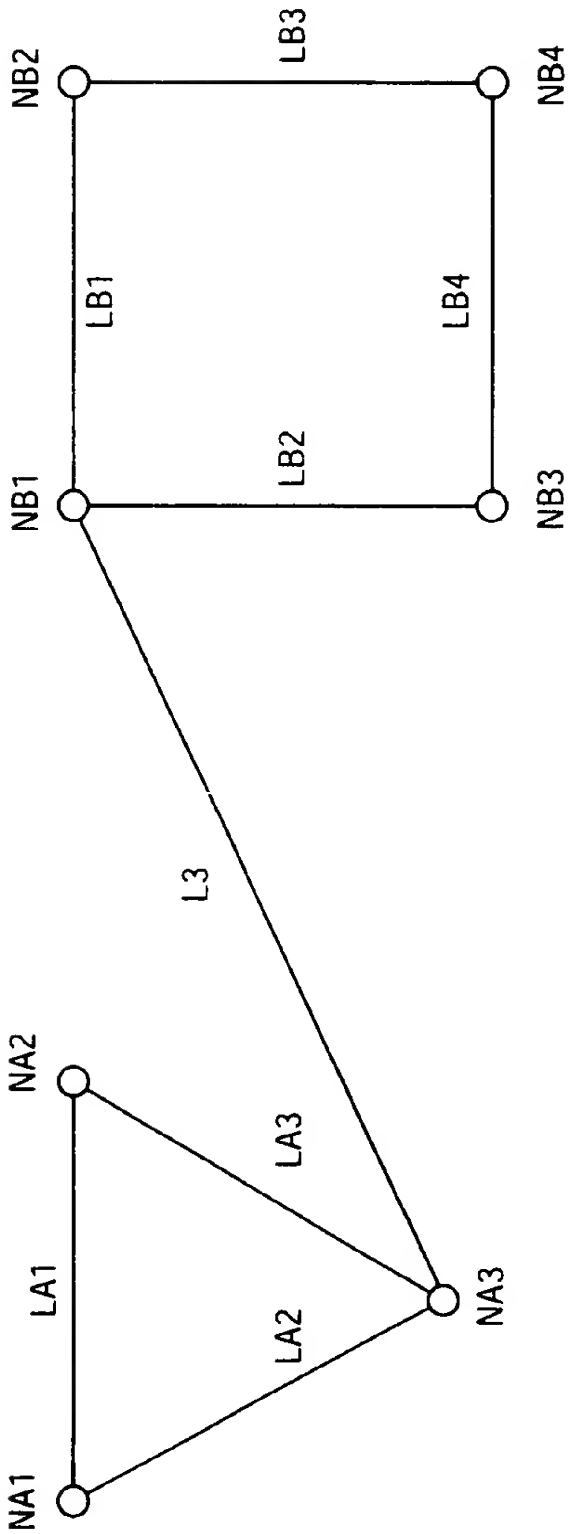



FIG. 3

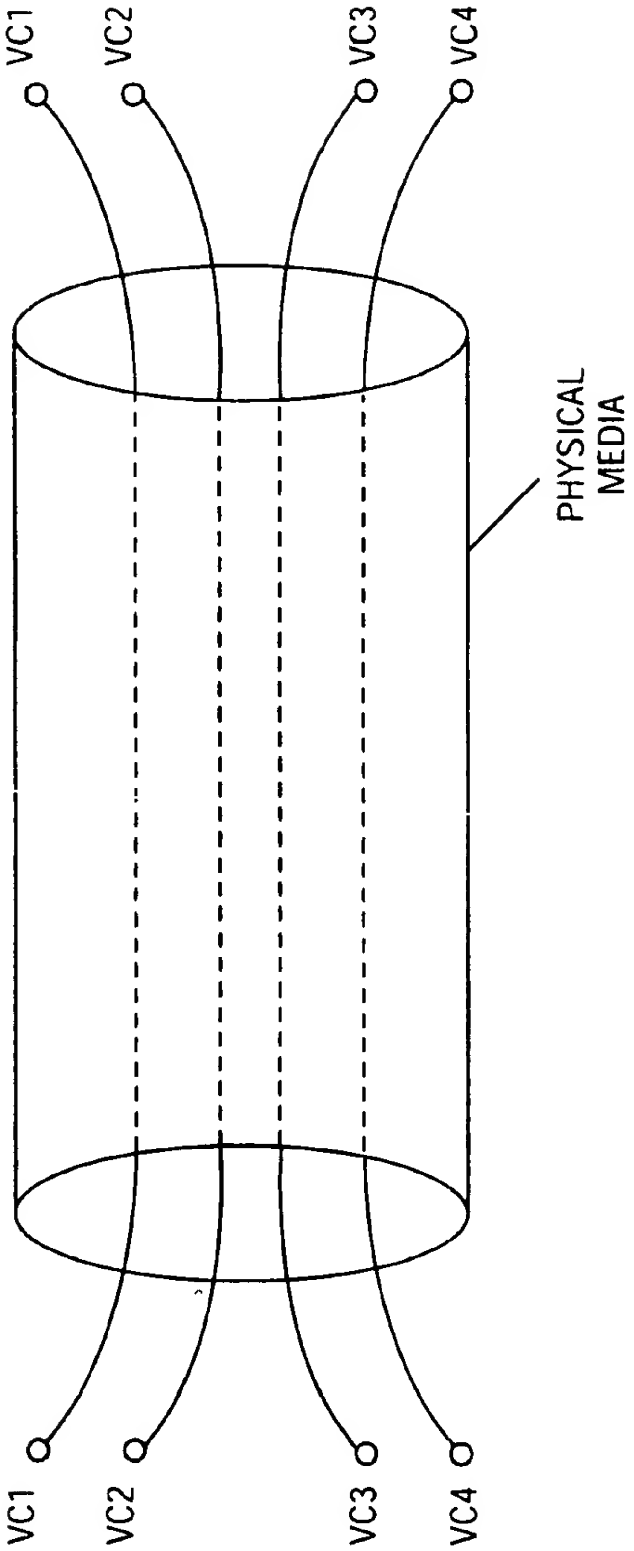



FIG. 4

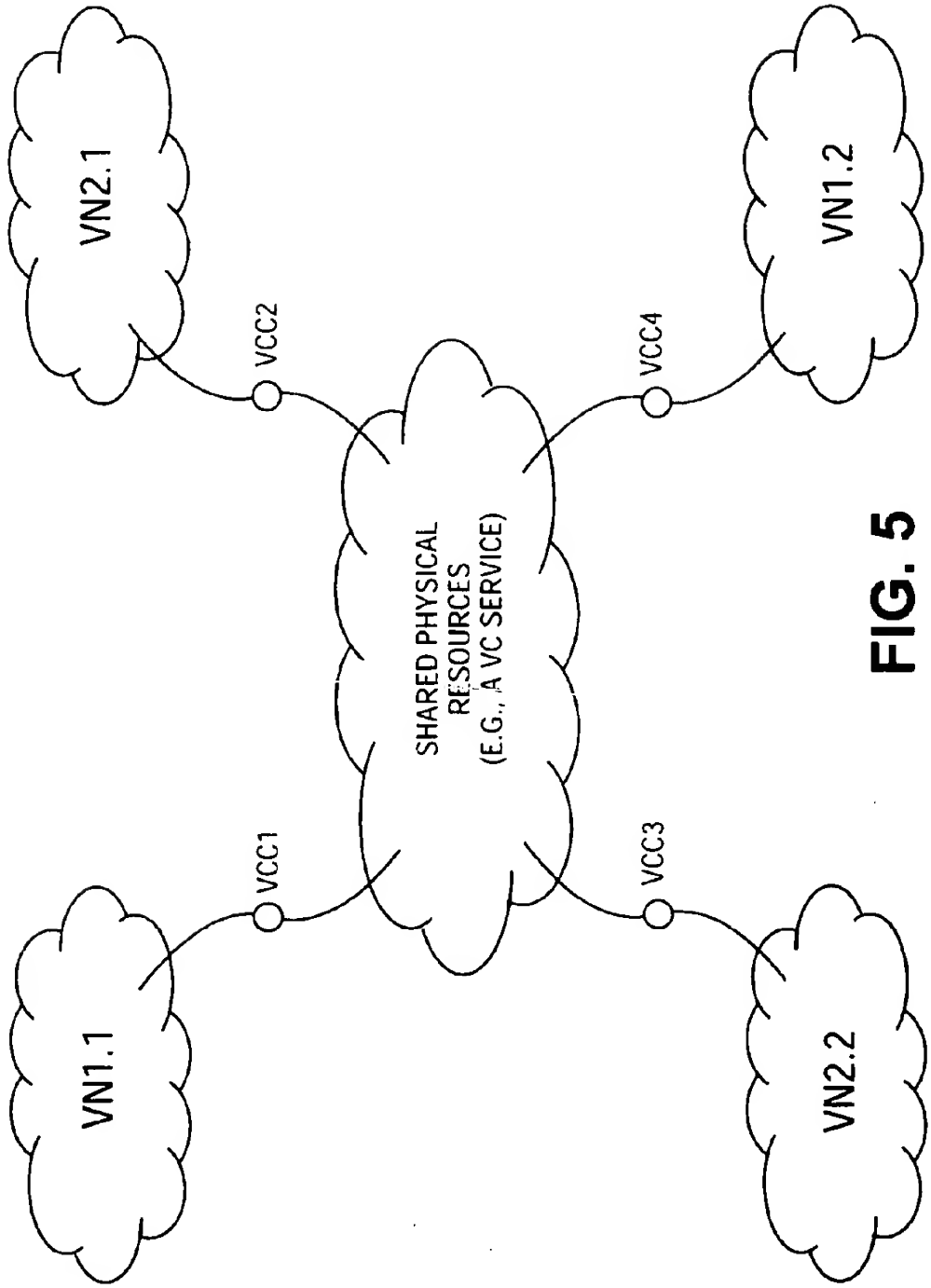

VN2.1
VN1.2
VCC2
VCC4
SHARED PHYSICAL RESOURCES (E.G., A VC SERVICE)
VCC1
VCC3
VN1.1
VN2.2

FIG. 5

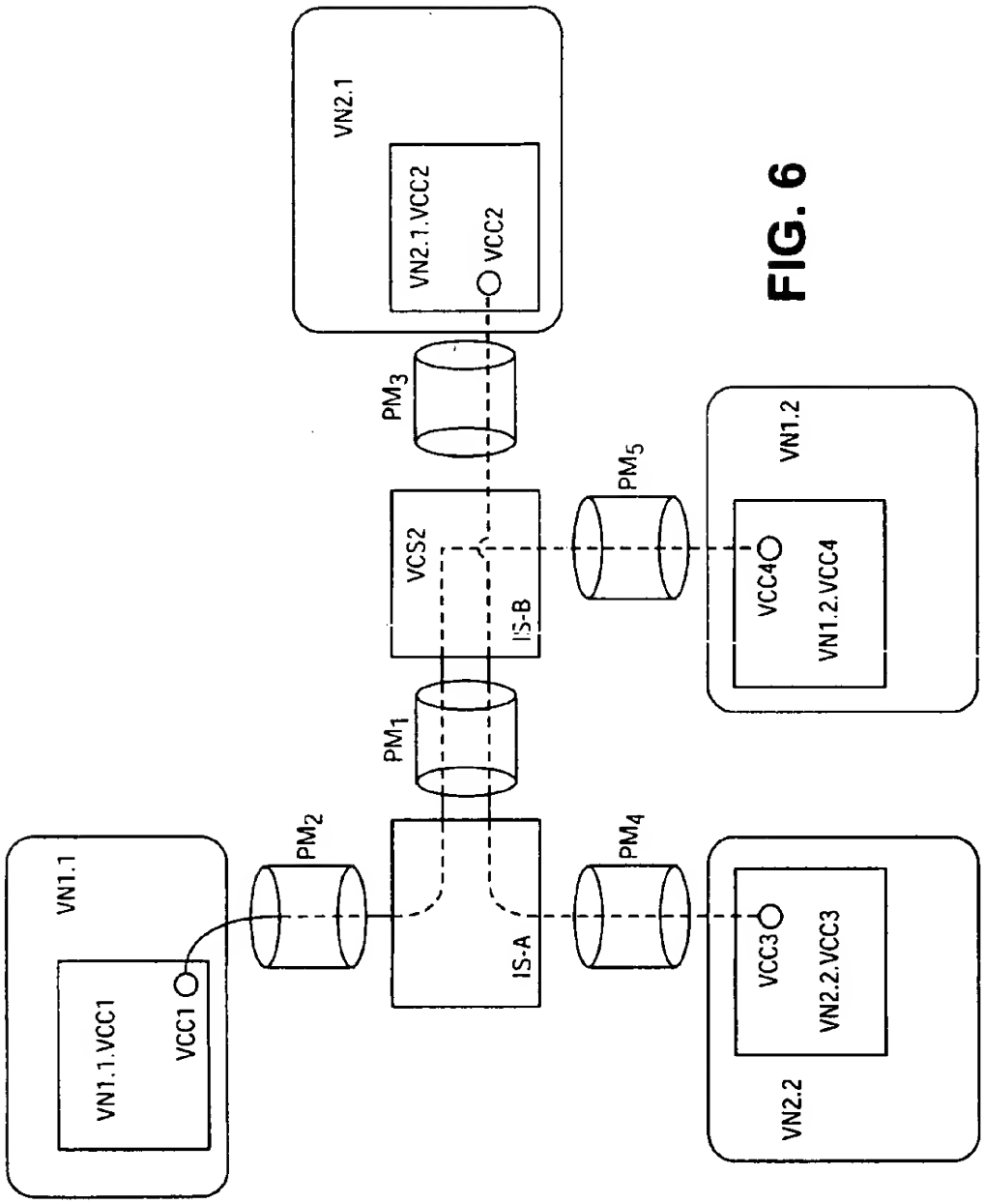
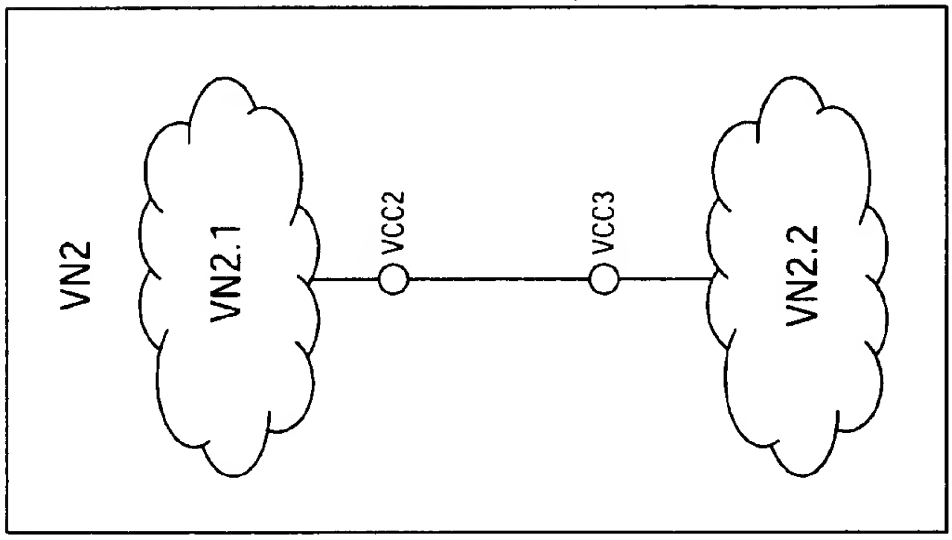

VN2.1
VN2.1.VCC2
VCC2
PM3
PM5
VN1.2
VCS2
IS-B
VCC4
VN1.2.VCC4
PM1
PM2
PM4
VN1.1
IS-A
VCC3
VN2.2.VCC3
VN1.1.VCC1
VCC1
VN2.2

FIG. 6



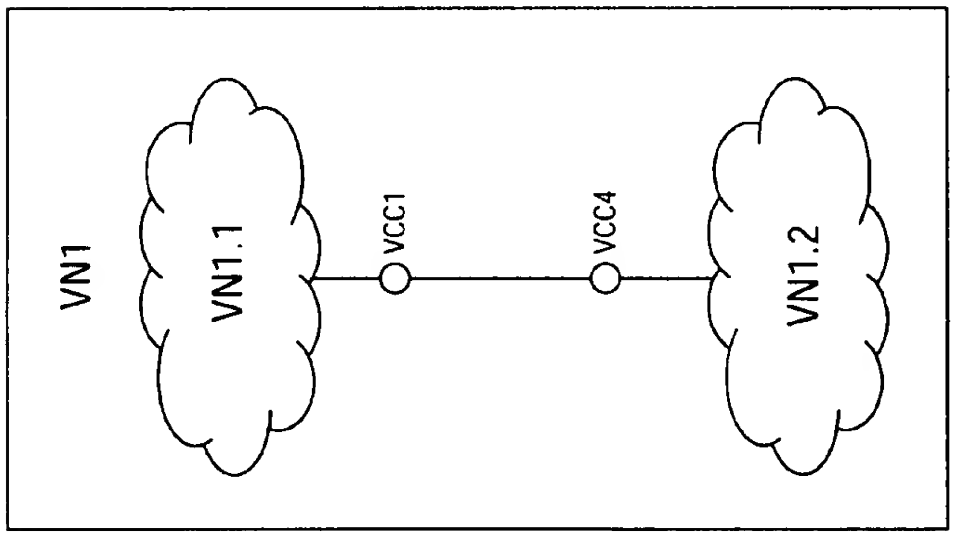




FIG. 7

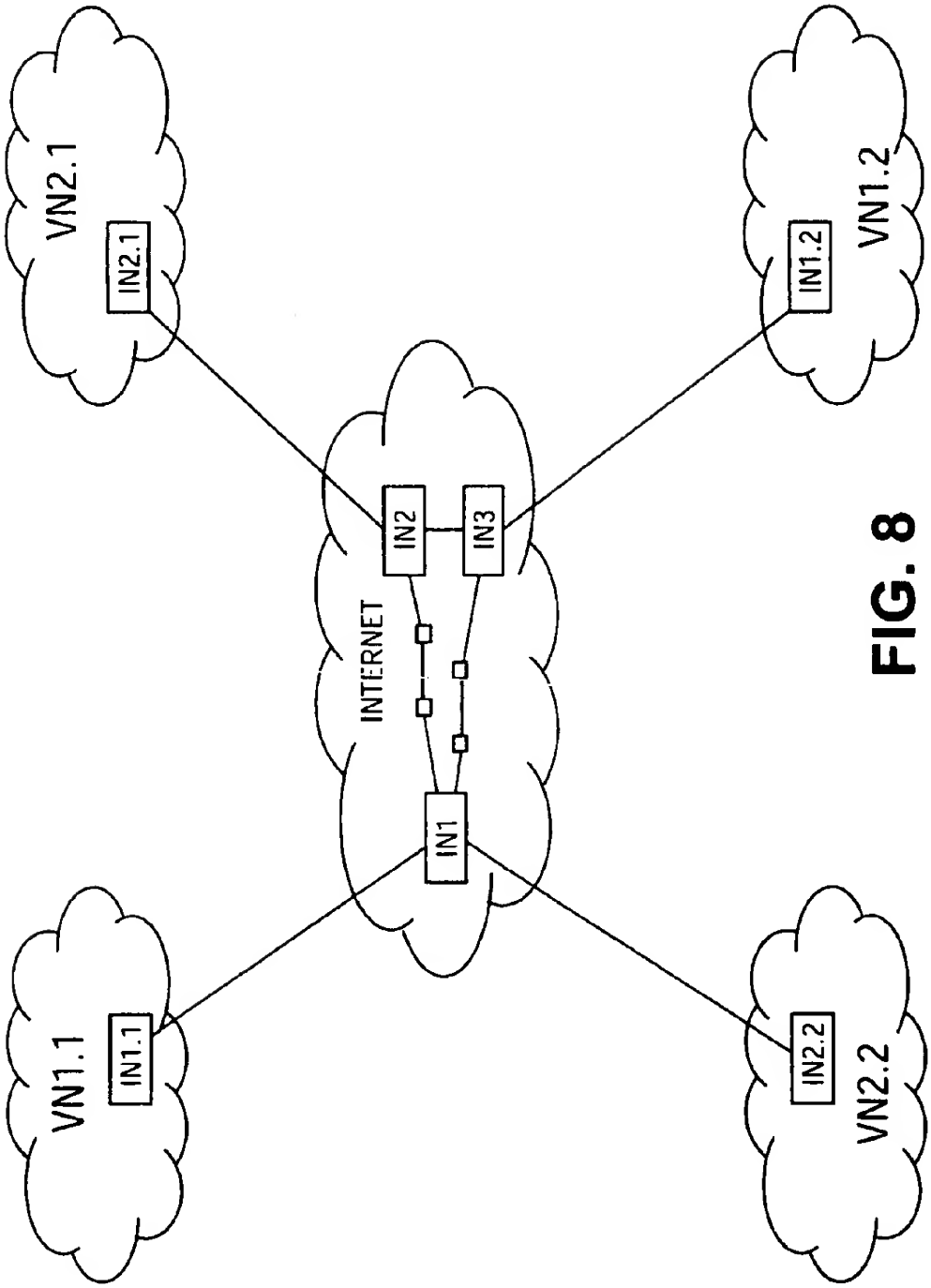



FIG. 8

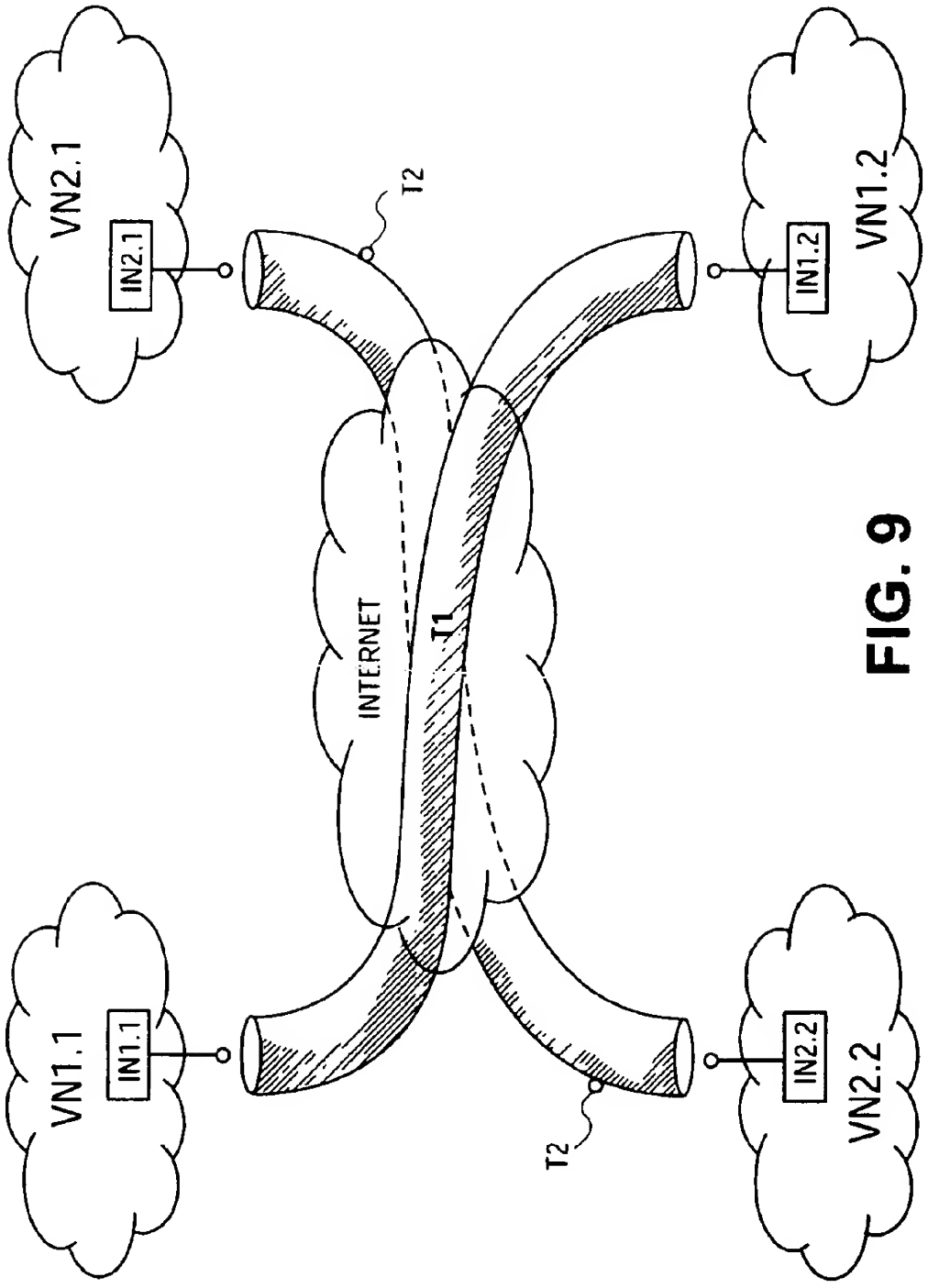



FIG. 9

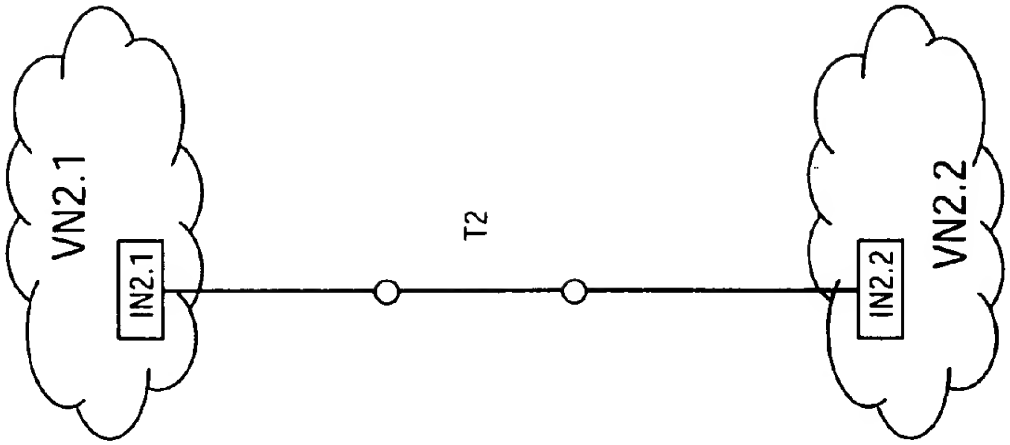
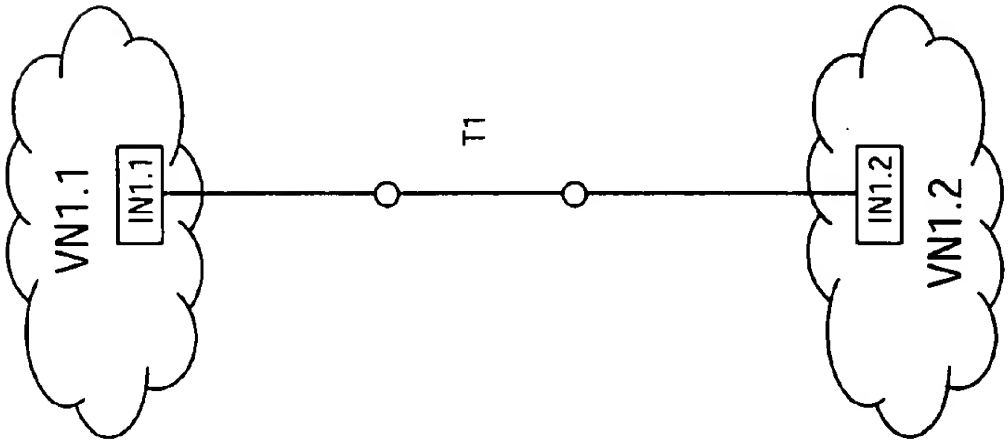



FIG. 10



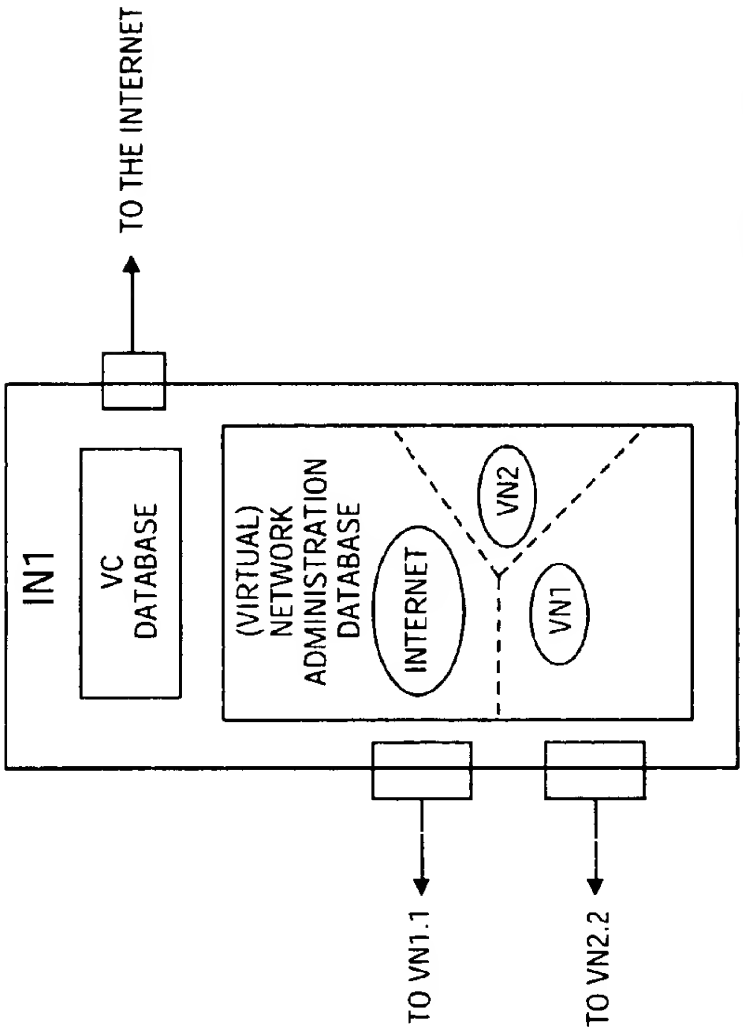

TO THE INTERNET
IN1
VC DATABASE
(VIRTUAL) NETWORK ADMINISTRATION DATABASE
INTERNET
VN2
VN1
TO VN1.1
TO VN2.2

FIG. 12

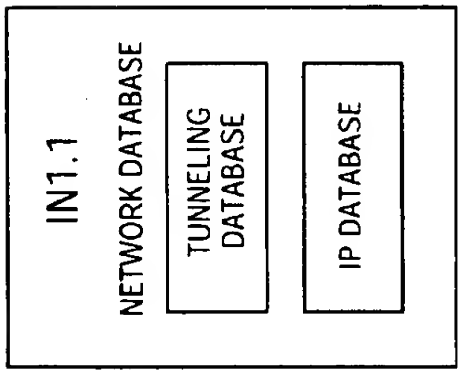

IN1.1
NETWORK DATABASE
TUNNELING DATABASE
IP DATABASE

FIG. 11

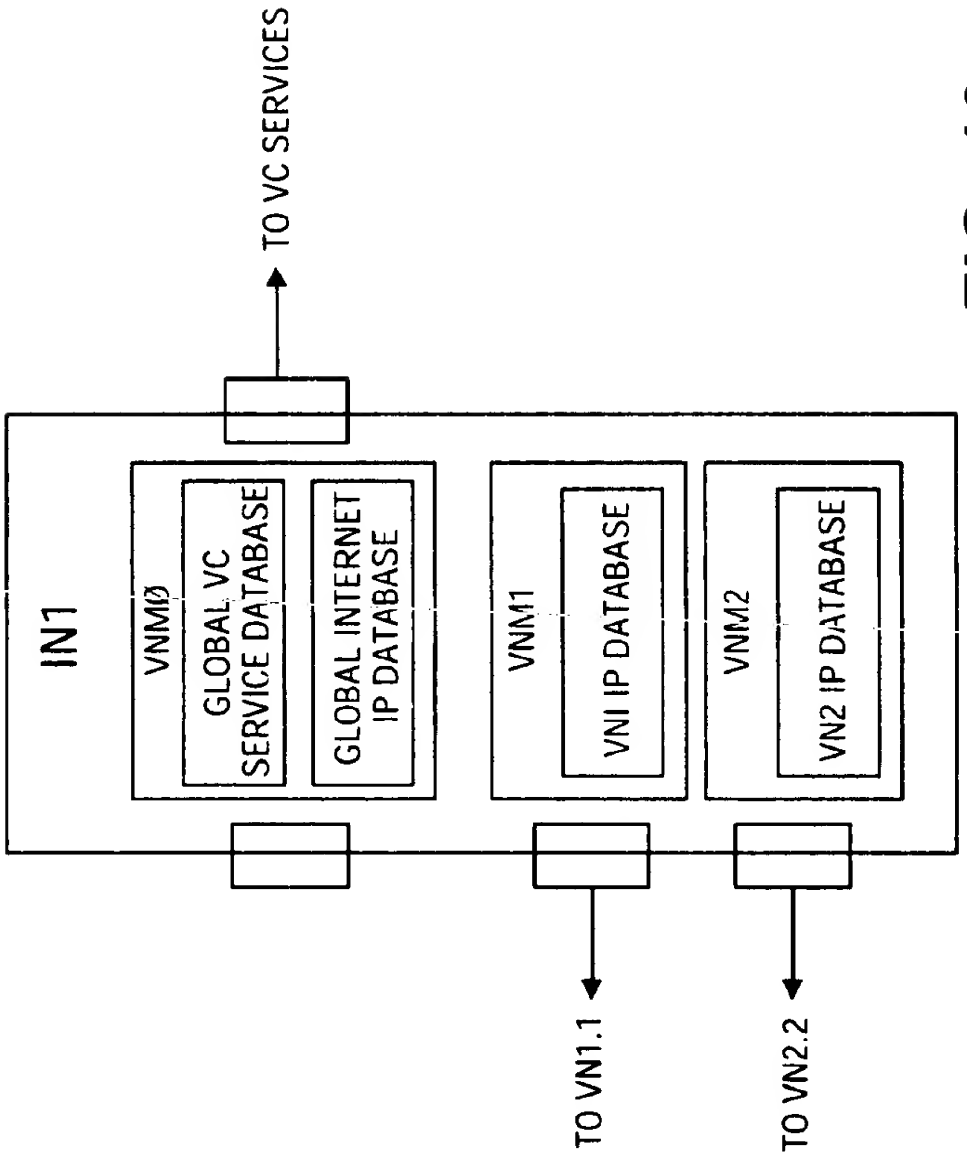

TO VC SERVICES
IN1
VNMØ
GLOBAL VC SERVICE DATABASE
GLOBAL INTERNET IP DATABASE
VNM1
VN1 IP DATABASE
VNM2
VN2 IP DATABASE
TO VN1.1
TO VN2.2

FIG. 13

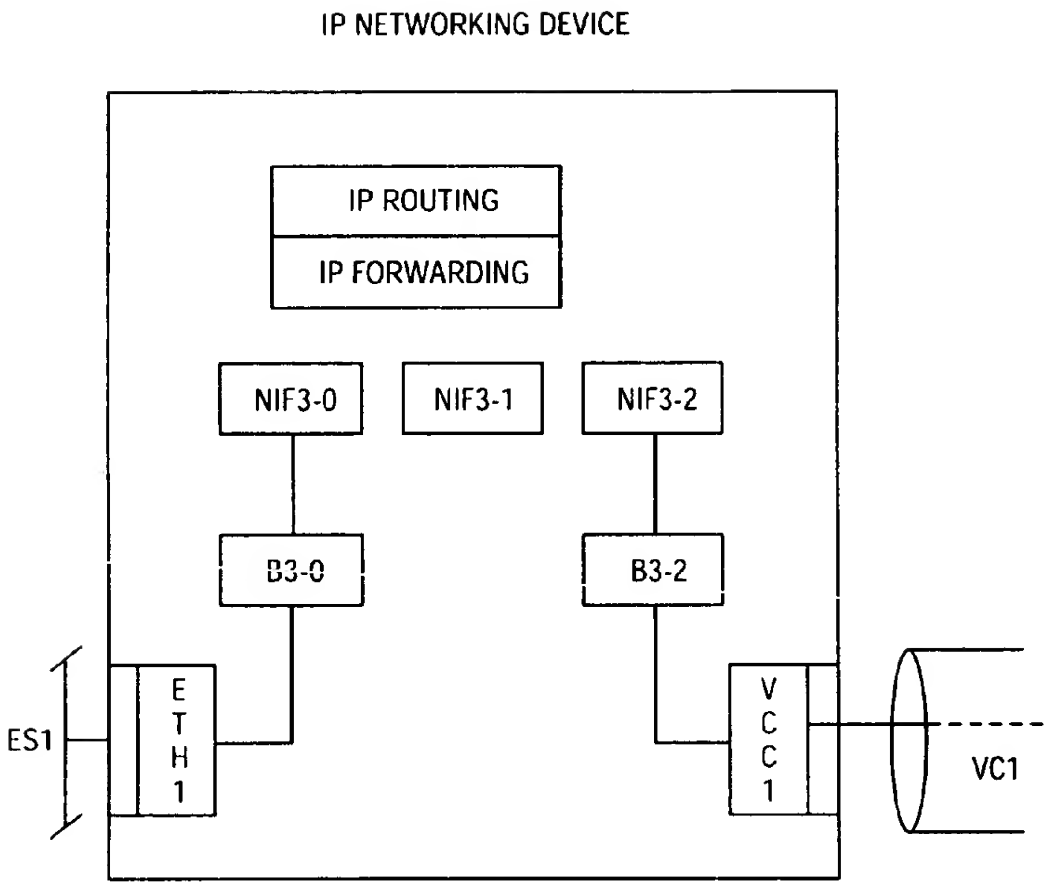

IP NETWORKING DEVICE
IP ROUTING
IP FORWARDING
NIF3-0
NIF3-1
NIF3-2
B3-0
B3-2
ES1
ETH1
VCC1
VC1

FIG. 14

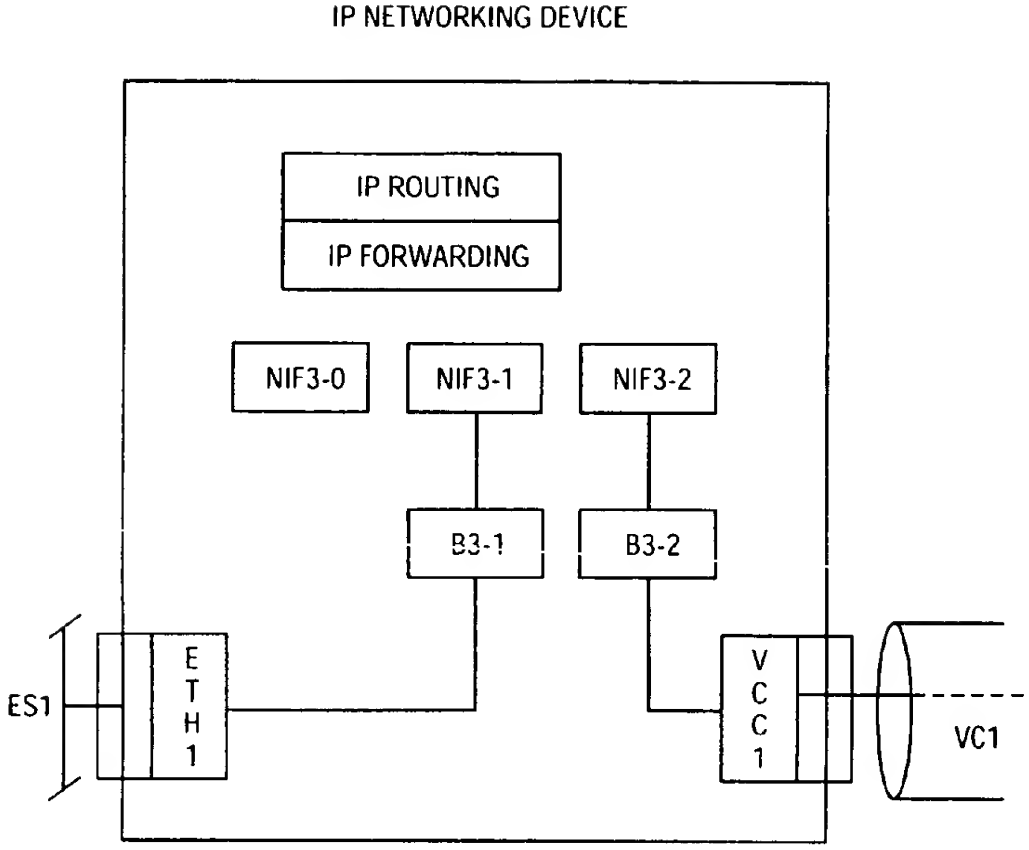

IP NETWORKING DEVICE
IP ROUTING
IP FORWARDING
NIF3-0
NIF3-1
NIF3-2
B3-1
B3-2
ES1
ETH1
VCC1
VC1

FIG. 15

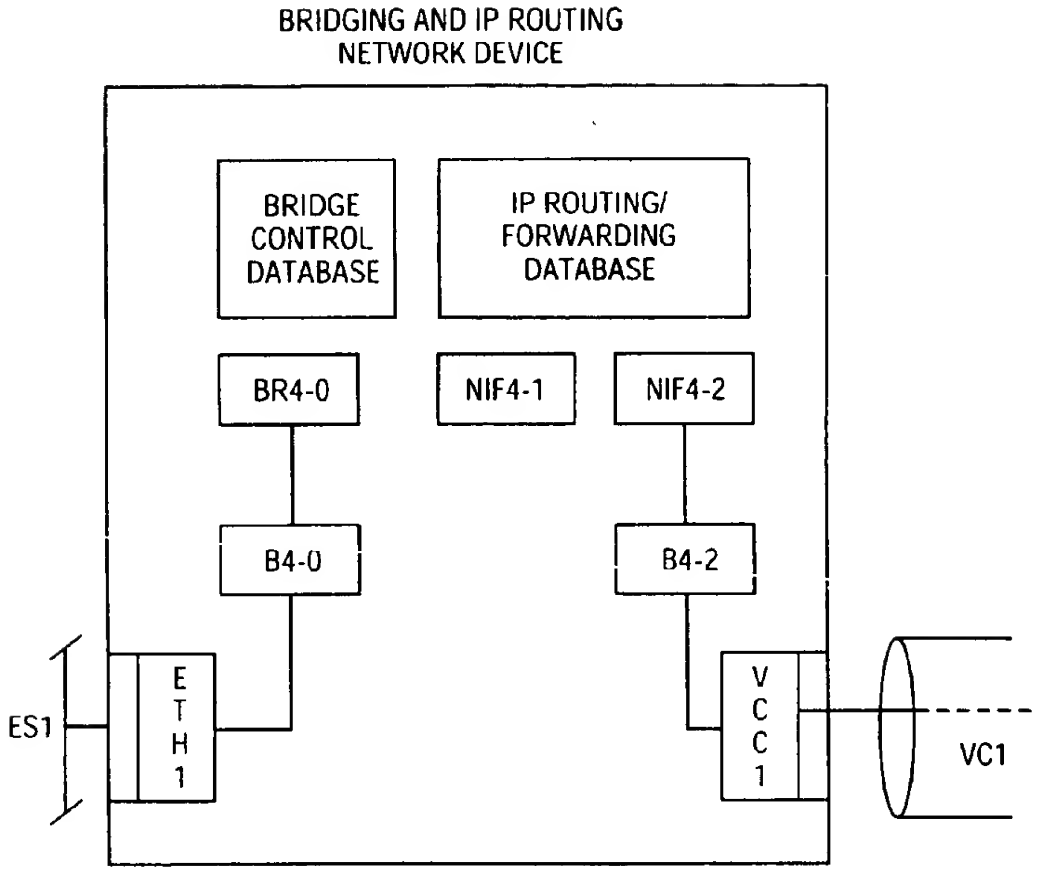

BRIDGING AND IP ROUTING
NETWORK DEVICE
BRIDGE CONTROL DATABASE
IP ROUTING/ FORWARDING DATABASE
BR4-0
NIF4-1
NIF4-2
B4-0
B4-2
ES1
ETH 1
VCC 1
VC1

FIG. 16

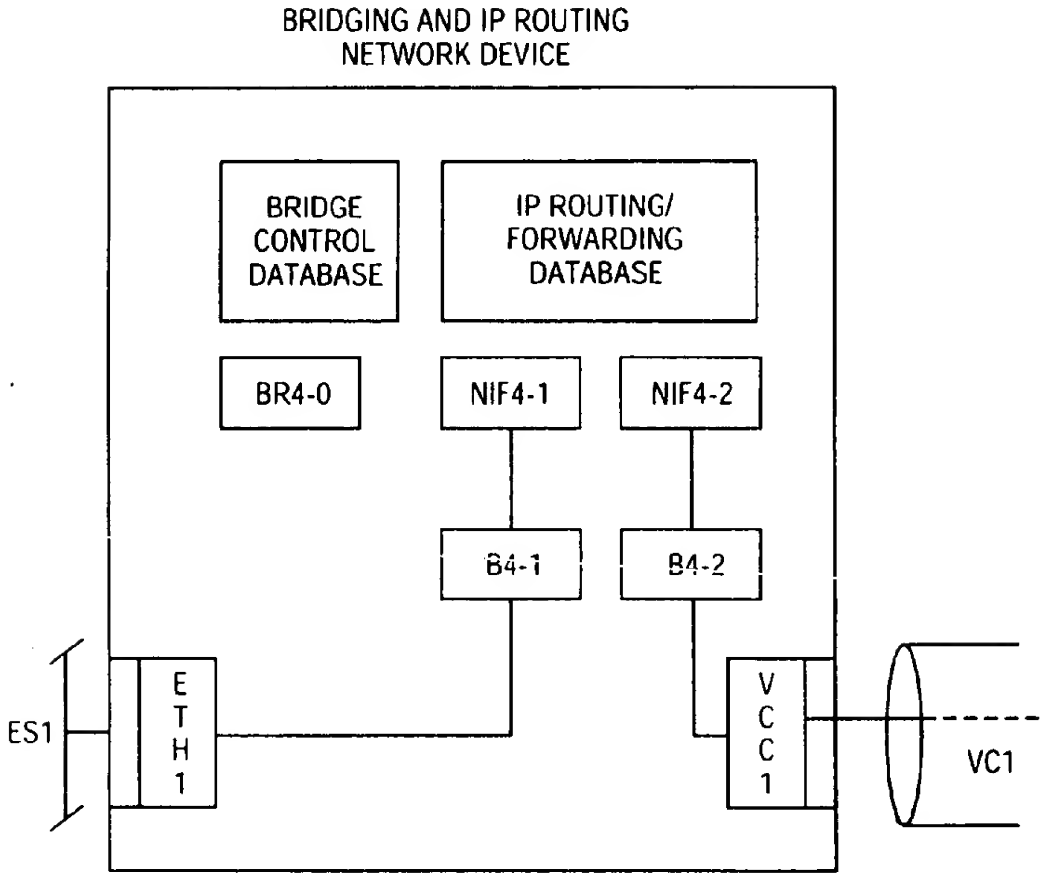

BRIDGING AND IP ROUTING
NETWORK DEVICE
BRIDGE CONTROL DATABASE
IP ROUTING/ FORWARDING DATABASE
BR4-0
NIF4-1
NIF4-2
B4-1
B4-2
ES1
ETH1
VCC1
VC1

FIG. 17

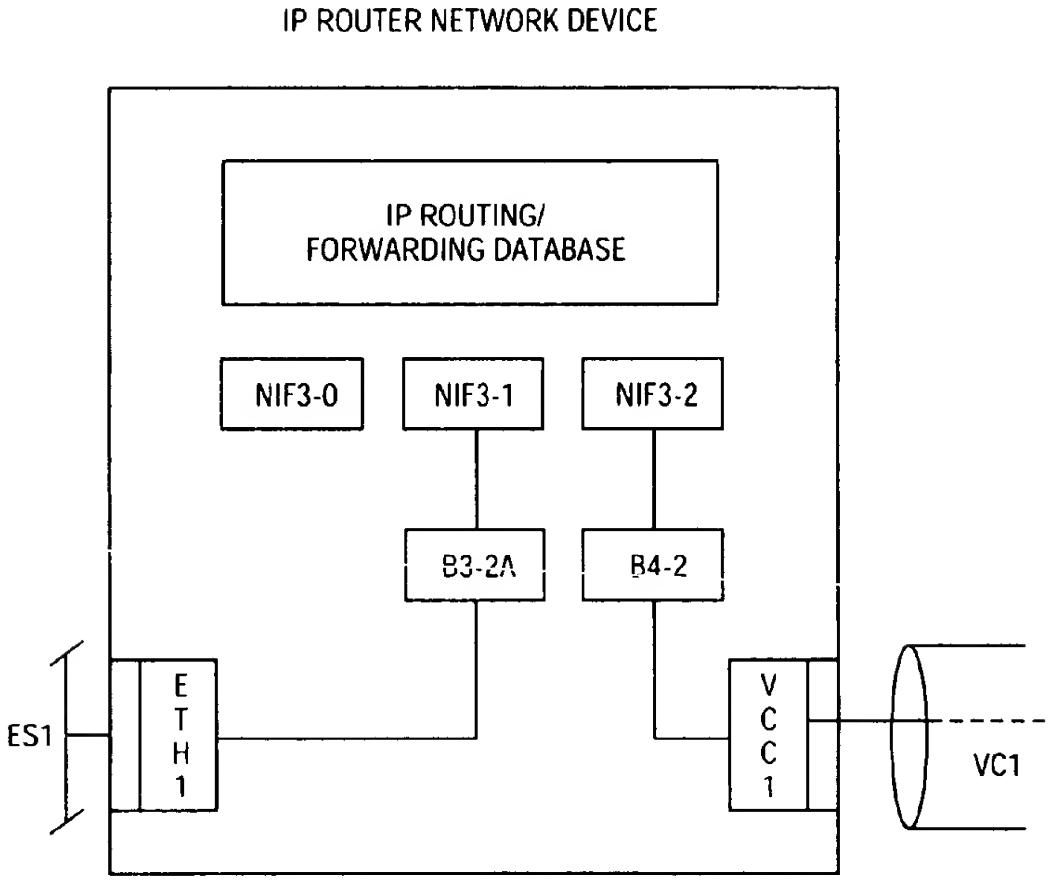

IP ROUTER NETWORK DEVICE
IP ROUTING/
FORWARDING DATABASE
NIF3-0
NIF3-1
NIF3-2
B3-2A
B4-2
ES1
ETH1
VCC1
VC1

FIG. 18

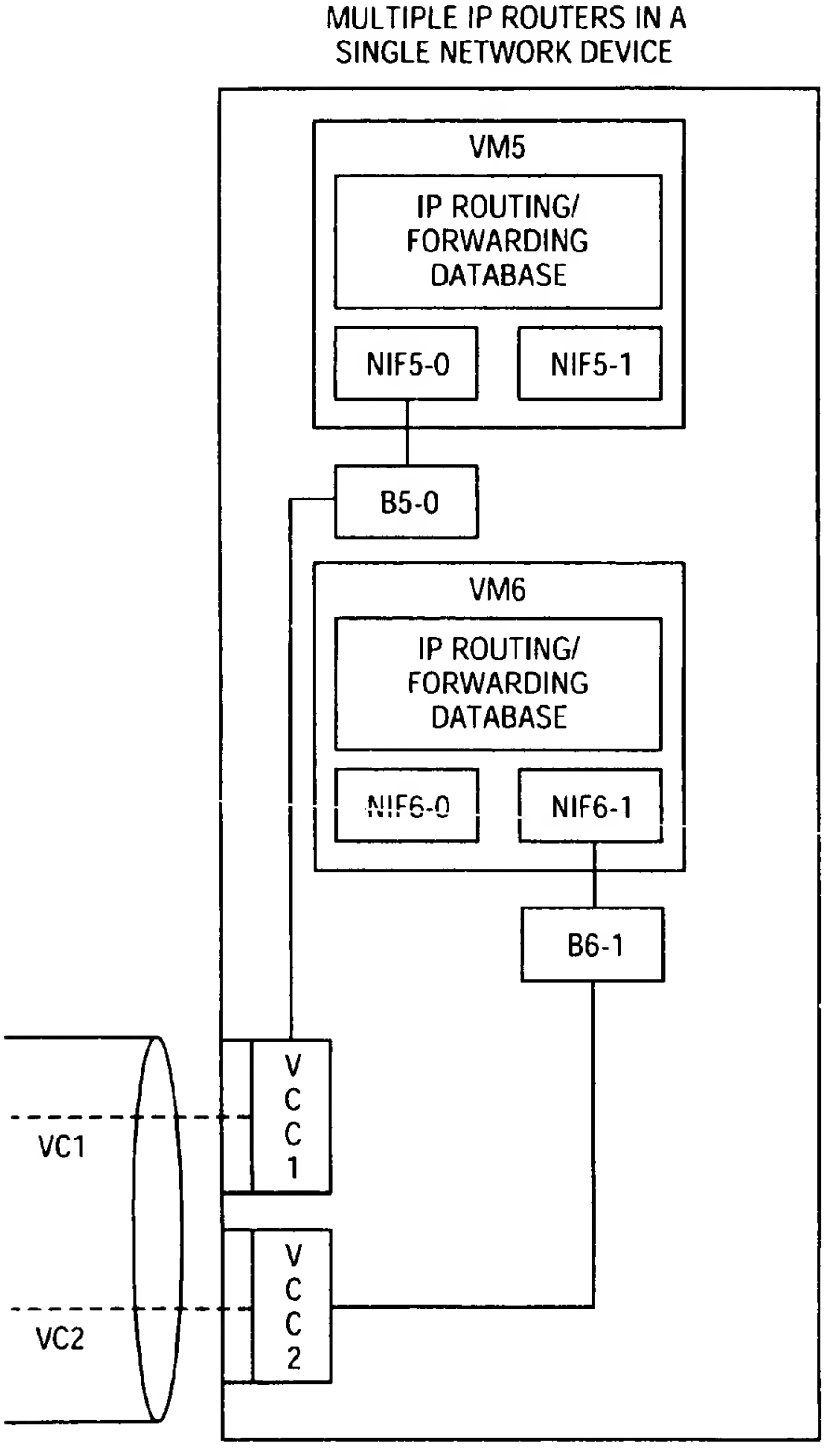

MULTIPLE IP ROUTERS IN A SINGLE NETWORK DEVICE
VM5
IP ROUTING/ FORWARDING DATABASE
NIF5-0
NIF5-1
B5-0
VM6
IP ROUTING/ FORWARDING DATABASE
NIF6-0
NIF6-1
B6-1
VCC1
VCC2
VC1
VC2

FIG. 19

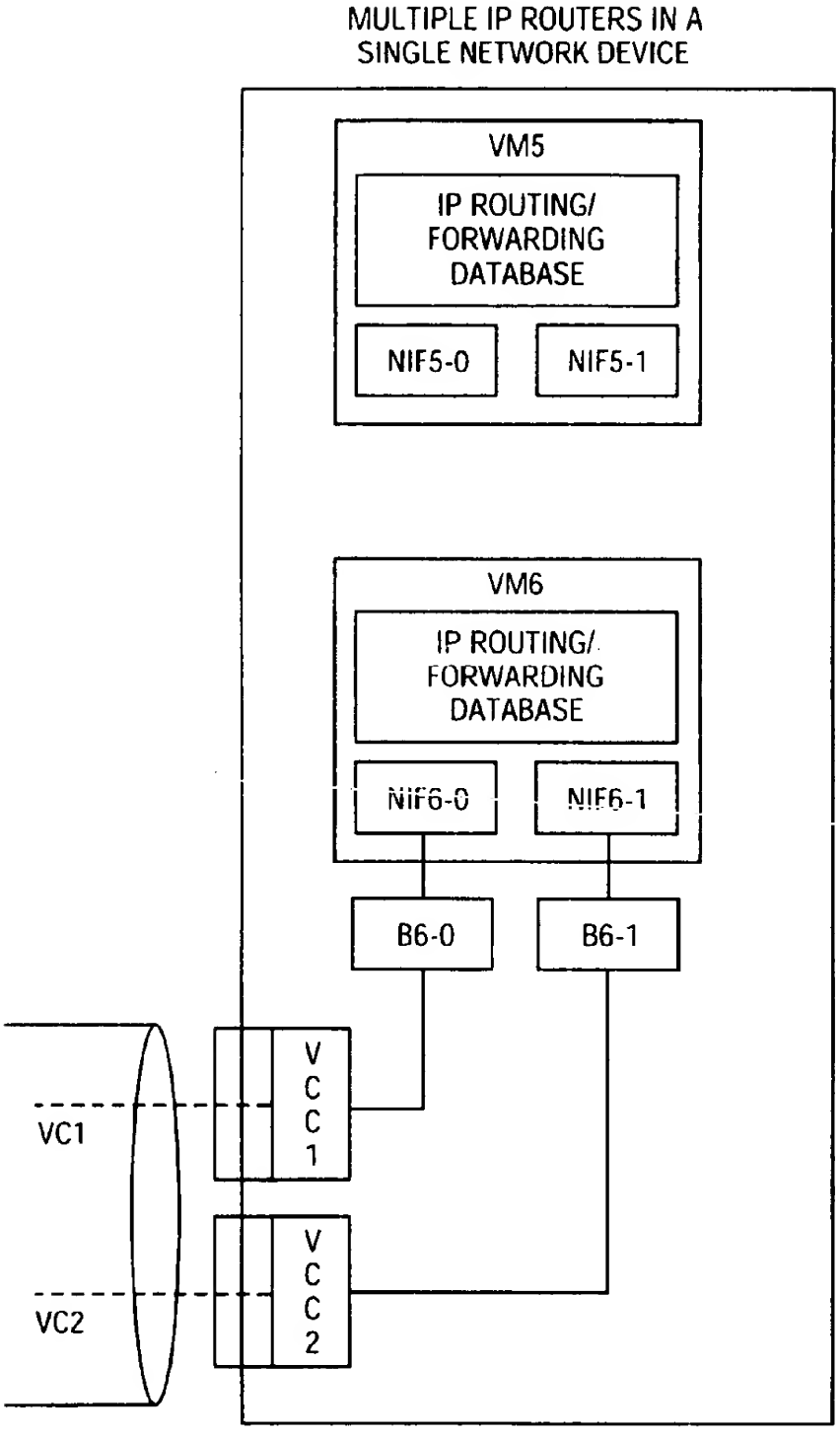

MULTIPLE IP ROUTERS IN A SINGLE NETWORK DEVICE
VM5
IP ROUTING/ FORWARDING DATABASE
NIF5-0
NIF5-1
VM6
IP ROUTING/ FORWARDING DATABASE
NIF6-0
NIF6-1
B6-0
B6-1
VCC1
VCC2
VC1
VC2

FIG. 20

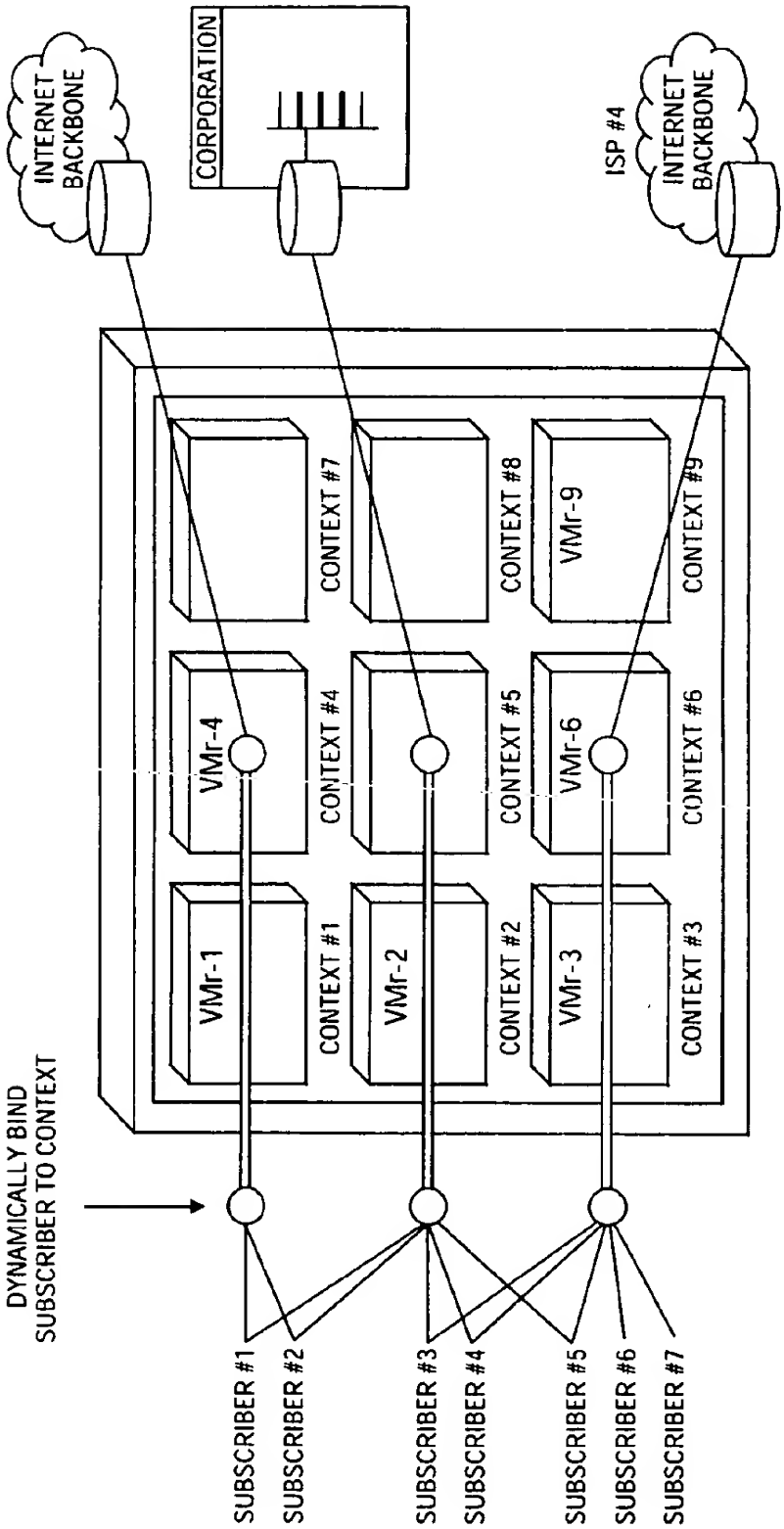

ISP #2
INTERNET BACKBONE
CORPORATION
ISP #4
INTERNET BACKBONE
CONTEXT #7
CONTEXT #8
VMr-9
CONTEXT #9
VMr-4
CONTEXT #4
CONTEXT #5
VMr-6
CONTEXT #6
VMr-1
CONTEXT #1
VMr-2
CONTEXT #2
VMr-3
CONTEXT #3
DYNAMICALLY BIND SUBSCRIBER TO CONTEXT
SUBSCRIBER #1
SUBSCRIBER #2
SUBSCRIBER #3
SUBSCRIBER #4
SUBSCRIBER #5
SUBSCRIBER #6
SUBSCRIBER #7

FIG. 21

# DOMAIN ISOLATION THROUGH VIRTUAL NETWORK MACHINES

## BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates in general to communications networks, and more particularly, to the operation of network devices that can operate in multiple virtual networks simultaneously.

2. Description of the Related Art

### Network Layering and Protocols

A communication network provides information resources transfer services that transfer information resources among devices attached to the network. Information resources, as the term is used herein, includes any form of information that can be transmitted over a network for use by or with any end station or network device connected to the network. Information resources, for example, may include computer programs, program files, web pages, data, database information, objects, data structures, program icons, graphics video information or audio information. *Computer Networks and Internets*, Douglas E. Comer, Prentice Hall, 1997, provides extensive information about communication networks.

Networks are built from devices or stations called nodes, and the communications channels that interconnect the nodes, called links. A set of nodes and links under one administrative authority is called a network domain. Communication between end stations attached to a network ordinarily is achieved through the use of a set of layered protocols. These protocols are generally described by reference to the Open Systems Interconnection (OSI) computer communications architecture. The standard OSI architecture includes seven layers: application, presentation, session, transport, network, data link and physical. A communication network may employ fewer than the full seven layers. However, the layer 2 and the layer 3 software protocols ordinarily play a prominent role in the transfer of information between interconnected networks and between end stations connected to the networks.

The physical layer is the lowest layer (layer 1) of the OSI model. There are numerous technologies that can be employed to build networks at layer 2. Layer 2 networks can be "connection oriented", meaning that a connection must be established before data can flow between two stations; ATM, Frame Relay, and X.25 are examples of connection oriented layer 2 protocols. Layer 2 networks can also be connection-less, meaning data can be transmitted without establishing any connection in advance; Ethernet and FDDI are two examples of connection-less layer 2 protocols.

In order to provide services useful to end users, the devices in a network must perform higher layer functions to create what are called "virtual networks". The "Internet" is one example of a very popular and public virtual network. The Internet uses the IP protocol to provide the higher layer (layer 3) functions required to support operation of the virtual network. There are many other private (virtual) networks that also uses the IP protocol. The term "internet" with a small "i" is used to differentiate between these less well known private internets, and the very popular and public large "I" Internet. There are many other protocols that can be used to construct virtual networks at layer 3, including IPX, DECnet, AppleTalk, CLNP, etc. There are many other private and public networks using these other layer 3 protocols, either independent of or in conjunction with the IP protocol.



Thus, networks can be built at many different layers. Each layer has its own function and its own type of nodes and links. Higher layer networks are built "on top of" lower layer networks. In other words, nodes at a given layer may use the services of the next lower layer to provide links for communication with peer nodes (i.e. nodes at the same layer on other devices). Routers are examples of nodes in a layer 3 network. Bridges are examples of nodes in layer 2 networks.

### Network Domains

A network domain as the term is used herein refers to the set of nodes and links that are subject to the same administrative authority. A single administrative authority may administer several networks in separate domains, or several layers of the same network in a single domain, or any combination. There are actually several possible administrative domains in any large virtual network. The boundaries of a network domain can be defined along the lines dividing layers of the protocol stacks. For instance, the same layer 1 physical devices and physical connections may have several layer 2 network domains layered onto them. These layer 2 domains, in turn, may have one or more layer 3 domains layered on top of them. A network domain may even transcend the boundaries between layers such that a layer 2 network and a layer 3 network may be part of the same network domain.

The administration of even a single network domain can be quite complex. Virtual networks have administrative authorities associated with them to control their higher layer functions. The cost of administering a network, physical or virtual, can be enormous, and is often the largest cost item in the operations of a network.

When several virtual networks are layered on top of the same layer 2 service or another virtual network, the boundaries between network domains may be somewhat obscure. The boundaries between the domains of the overlaid virtual networks intersect at points where they must share physical or virtual resources. In practice, the administrators of the overlaid virtual networks are very concerned about sharing resources, especially when they are competing commercial entities. Concerns arise about integrity, privacy, and security of data and network control information flowing across the shared resources at the lower layers. The administrators of the underlying networks are called upon to solve complex administrative problems. The costs of administering these networks increases quickly with the number of virtual networks, their size, the complexity and compatibility of their individual policies, and increased demands for security, integrity, and isolation between domains.

### Network Devices and Databases

The term network device is used here to refer to the collection of mechanisms (e.g. computer and communications hardware and software) used to implement the functions of a station in a network. A network device contains some capacity to store and operate on information in databases in addition to the ability to transmit and receive information to and from other devices on the network. Examples of network devices include but are not limited to routers, bridges, switches, and devices that perform more than one of these functions (e.g. a device that does both routing and bridging).

A router is an example of a network device that serves as an intermediate station. An intermediate station is a network

device that interconnects networks or subnetworks. A typical router comprises a computer that attaches to two or more networks and that provides communication paths and routing functions so that data can be exchanged between end stations attached to different networks. A router can route packets between networks that employ different layer 2 protocols, such as Token Ring, Ethernet or FDDI, for example. Routers use layer 3 protocols to route information resources between interconnected networks. Nothing precludes a network device that operates as an intermediate station from also operating as an end station. An IP router for example typically also operates as an end station.

A router can understand layer 3 addressing information, and may implement one or more routing protocols to determine the routes that information should take. A multiprotocol 10 router runs multiple layer 3 protocols such as IP, IPX or AppleTalk for example. A router also be characterized as being multiprotocol if it runs multiple adaptive routing protocols such as RIP, BGP or OSPF all feediing a single IP layer.

The network device router configuration of FIG. 1A depicts what is often referred to in industry as a multiprotocol bridge/router. In this illustrative example, there are separate databases for three layer 2/3 networking protocols: bridging, IP routing, and IPX routing. The example IP database employs both the OSPF and RIP dynamic routing protocols. Thus, the intermediate station node of FIG. 1A includes both multiple networking protocols and multiple routing protocols.

A bridge is another example of a network device that serves as an intermediate station. A typical bridge comprises a computer used to interconnect two local area networks (LANs) that have similar layer 2 protocols. It acts as an address filter, picking up packets from one LAN that are intended for a destination on another LAN and passing those packets on. A bridge operates at layer 2 of the OSI architecture.

The term network database will be used to refer to all the control information housed in a network device required to support the device's operation in a set of one or more networks. Each device in a network holds its own network database. In order for the network at large to operate properly, the network databases of all network devices in a network domain should be consistent with each other. The network database control information defines the behavior of its network device. For example, not only might it determine whether the network device will function as a router or a bridge or a switch, but also it will determine the details of how the device will perform those functions.

When a network device is deployed to operate in multiple domains, its network database can become quite complex. The cost of administering the network device increases significantly when the network database is more complex. The cost of administration is already the most significant cost of operating many networks, and the trend toward greater complexity through greater use of virtual networking continues unabated.

The information found in a typical network database includes, but is not limited to, data used to configure, manage, and or monitor operations of:

Communications Hardware (e.g. layer 1 transceivers/drivers/chips etc.)

Computer Hardware

Computer Software

Layer 2 Addressing

Layer 2 Connections (Layer 2 interfaces)

Traffic filter policies

Bridging (IEEE 802.ID)

Bridge filters and or policies

Network (layer 3) Addressing

Layer 3 Connections (Layer 3 interfaces)

(Network/layer 3) Address Translation (NAT) policies

Access Control (e.g. user names and password)

Access policies (e.g. what user can use what services)

Routing (IETF RFC 1812)

Routing Protocols (e.g., BGP, OSPF, RIP, IGRP, etc.)

Route filters and policies (e.g. route leaking)

Tunneling

Tunneling Protocols (e.g., L2TP, GRE, PPTP, etc.)

A single network device can operate in one or more (virtual) network domains. For each domain in which a device operates, it needs to store information about that domain in some database form.

Much of the information in a network database must be configured manually; particularly the policy information as it must reflect the administrator's subjective wishes for how the network should operate. Manual configuration involves human effort, which can become expensive, especially as the number of policies and their complexity increases. Network administrative chores include the assignment of user names, passwords, network addresses or other user identifiers, and configuration of policy databases. This configuration and management may be used to establish traffic filtering policies such as what kind of information payloads will be carried. Traffic and Route filtering policies may be established to determine what paths through the network will be used for each payload carried. Access control policies may be to dictate which users at which end stations have access to which services at other end stations. Security policies may be established to ensure the integrity of the information payloads. Each configured bit of policy somehow finds its way into the network database of the device implementing the policy.

*Cisco Router Configuration* by A. Leinwand, B. Pinsky and M. Culpepper, published by MacMillan Technical Publishing, Indianapolis, Ind., 1998 provides an extensive treatment of the configuration of the databases of Cisco System routers. This is just one example of a network device database.

## Building Virtual Networks

The layering of software protocols in accordance with the ISO architecture makes possible the creation of "virtual networks". Virtual networks are to be contrasted with physical networks. Two physical networks which have no physical devices or links in common, can be said to be physically isolated from each other. Physical isolation may be required in order to ensure that a network has the highest levels of security and integrity.

Physical networks are defined at layer 1 of the OSI model. Virtual networks, on the other hand, are created at higher layers. It is possible to create multiple virtual networks all sharing common physical resources. A network is definitely virtual if it shares a common physical medium or device, such as an intermediate station, with any other (virtual) network. There are many conventional technologies and many commercially available products which can be used to build many types of virtual networks. For example, virtual circuits are a layer 2 construct that can be employed to create virtual networks.

It has been common practice in the industry for phone companies to offer connection oriented layer 1 and 2 services to Internet Service Providers (ISPs), corporations, and residential customers. These customers may build one or more higher layer (layer 3 and above) virtual networks on top of such publicly available layer 1 and 2 services. The higher layer virtual networks share a common set of layer 1 and 2 services, each having it's private set of virtual circuits.

A PC or a server are examples of end stations. End stations located at home or business, for example, may connect into an internet through an internet service provider (ISP). There are regional, local and global ISPs. In most cases, local ISPs connect into the regional ISPs which in turn connect into other regional or national ISPs. FIG. 1B illustrates an example of a connections to an ISP. In the example, home user end stations may connect via modems over dial-up lines to an ISP's router or remote access server (RAS). This data link often runs the PPP (Point-to-Point Protocol) which encapsulates and delivers packets to the ISP's site. Business user end systems may connect to the ISP through leased lines such as T1 lines or T3 lines depending on bandwidth requirements for example. Other examples of typical connection options between home or business users and an ISP include ISDN, T1, fractional T1, various optical media, and xDSL. ISPs may also offer tunnel mode or transport mode services that help businesses set up virtual private networks (VPNs) between remote end stations and virtual dial-up services for remote and mobile end stations.

The ISP serves as a conduit for information transmitted between the end stations in the home and other end stations connected to the Internet.

A virtual circuit is a dedicated communication channel between two end stations on a packet-switched or cell-relay network. ATM, Frame Relay, and X.25 are all different types of virtual circuit based networking technologies. A virtual circuit follows a path that is programmed through the intermediate stations in the network.

There are permanent and switched virtual circuits. A permanent virtual circuit (PVC) is permanent in the sense that it is survives computer reboots and power cycles. A PVC is established in advance, often with a predefined and guaranteed bandwidth. A switched virtual circuit (SVC) is "switched" in the sense that it can be created on demand analogous to a telephone call. Both PVCs and SVCs are "virtual" circuits in that they typically are not allocated their own physical links (e.g. wires), but share them with other virtual circuits running across the same physical links.

"Tunneling" is one mechanism for building higher layer networks on top of an underlying virtual network. Tunneling has already gained acceptance in the industry and several technologies are either in operation or under development. Some of the tunneling protocols used in IP networks for example include L2TP, GRE, PPTP, and L2F. There are many other Tunneling technologies used in IP and other protocols.

Referring to FIGS. 2A–2B, there are shown network graphs representing two illustrative networks. Network A is represented by three nodes (NA1, NA2, and NA3), and three links (LA1, LA2, and LA3). Network B is represented by four nodes (NB1, NB2, NB3, and NB4) and four links (LB1, LB2, LB3, and LB4). As used herein, the term node may represent any end station or intermediate station, and the term link means any connection between nodes. If these are physical nodes and links, Networks A and B are physically isolated from each other. If these are virtual (circuit) links which actually depend on a shared physical medium, then the two (virtual) networks are said to be virtually isolated from each other.

Illustrative Networks A and B each may be part of different network domains. Independent administrative control may be exercised over each of the Network A and B domains, for example, through the configuration and management of intermediate stations such as bridges and routers.

Referring to FIGS. 2A and 2B, it will be appreciated that the independent administration of the Network A and Network B domains may result in incompatible policies as between the two domains. This is not a problem provided that the domains remain isolated from each other. Referring to FIG. 3, however, there is shown a network graph of Network C which comprises Networks A and B joined by link LJ. The isolation between Networks A and B, whether physical or virtual, is lost when they are joined in Network C. This joining of the two Networks A and B may create challenges to the administration of combined Network C. For example, despite the joining of the two networks, there still may be a need to apply different or even conflicting policies to each of Networks A and B. In essence, the administrative challenge is to maintain the administrative integrity of the Network A domain and the administrative integrity of the Network B domain despite the fact that both of these networks are part of Network C and are no longer physically isolated from each other.

FIG. 4 is an illustrative drawing of a segment of a single physical medium capable of carrying multiple information flows, each in its own virtual circuit (or channel). The physical medium may for instance be a cable or a wire or an optical fiber. The segment shown is carrying four independent information flows on four different virtual circuits; VC1, VC2, VC3, and VC4. These virtual circuits, for example, may be implemented using X.25, ATM, Frame Relay, or some other virtual circuit (or channelized) service.

FIG. 5 is an illustrative drawing representing an example of two virtual networks (VN1, and VN2) each made up of two independent network segments (VN1.1 and VN1.2 for VN1, and VN2.1 and VN2.2 for VN2). All segments connect to shared physical network resources. In this example, the shared network resources of FIG. 5 provide a virtual circuit service. A virtual circuit connection to an end station or intermediate station connection to a virtual circuit is called a virtual channel connection (VCC). VN1 connects at VCC1 and VCC4; and VN2 connects at VCC2 and VCC3. The shared network resources also provide virtual circuit service that connect VCC1 and VCC4 so as to join VN1.1 and VN1.2 into VN1 and so as to join VN2.1 and VN2.2 into VN2.

FIG. 6 is an illustrative drawing that provides additional details of some of the physical constituents of the virtual networks of FIG. 5. An intermediate station labeled VN1.1.VCC1 in VN1 connects segment VN1.1 to the VC service at VCC1. An intermediate station labeled VN1.2.VCC4 in VN1 connects segment VN2 to the VC service at VCC4. The VC service connects VCC1 to VCC4, linking VN1.1 to VN1.2 at the virtual circuit level. More specifically, physical media segments PM2, PM1 and PM5 and intermediate stations IS-A and IS-B provide the requisite physical infrastructure upon which the virtual circuit connection linking VN1.1 and VN1.2 is carried. This first virtual circuit connection serves as a network link between the VN1.1.VCC1 and VN1.2.VCC4 intermediate stations, to create one virtual network from the two segments VN1.1 and VN1.2.

Similarly, VCC2 and VCC3 are connected by the virtual circuit service, which connects intermediate stations

VN2.1.VCC2 and VN2.2.VCC3, joining the VN2.1 and VN2.2 segments to form the virtual network labeled VN2. More particularly, physical media segments PM4, PM1 and PM3 and intermediate stations IS-A and IS-B provide the virtual connection linking VN2.1 and VN2.2. The second virtual circuit connection serves as a network link between the VN2.1.VCC2 and VN2.2.VCC3 intermediate stations, to create one virtual network from the two segments VN2.1 and VN2.2.

FIG. 7 is an illustrative drawing shows the logical or higher level view of the two virtual networks VN1 and VN2 of FIGS. 5 and 6. It will be appreciated from the view of FIG. 6 that they share physical resources, and it will be appreciated from the view of FIG. 7 that they are logically or virtually separate.

In the illustrative example of FIG. 8, two virtual networks are layered on top of a third virtual network. The sharing of a common set of physical or virtual network resources by several virtual networks increases the challenges of maintaining isolation and security of the individual virtual networks. Nevertheless, end user requirements for information resources, technology advances, economics, politics, and regulations surrounding the networking industry are driving commercial, private and government entities to share common physical and virtual network infrastructure. Therefore, there are ever increasing demands imposed upon network administrators, and vendors of networking equipment.

In the illustrative drawing of FIG. 8, three separate network domains intersect at node IN1: i) that of the Internet itself (including or subsuming that of the underlying VC service supporting the Internet); ii) that of private virtual network VN1; and iii) that of private virtutal network VN2. This intersection of three network domains creates the potential for the kinds of administration and policy challenges discussed above. It will be noted that these networks are represented by different network "clouds" that symbolize the multifarious nodes and links in each of the networks.

The illustrative drawing of FIG. 8 illustrates an example of building two virtual networks on top of another virtual network similar to the previous example in FIGS. 5, 6 and 7. As before, the virtual networks being overlaid are each composed of two segments. Using a tunneling protocol or some other higher layer (layer 3 or above) mechanism, connections are made between nodes IN1.1 and IN1.2 to form a link to tie the two segments of VN1 together. This link is shown as T1 in FIG. 9 and 10. Link T2 is similar, formed between nodes IN2.1 and IN2.2, to tie the two segments of VN2 together. The logical view of the two virtual networks in FIG. 9 is shown in FIG. 10, which bears a very strong resemblance to FIG. 7. The important difference to note between the examples is that in FIG. 7 a layer 2 VC network was used as the underlying network shared resources, and in FIG. 10 another virtual network was used as the underlying network shared resources; specifically, a tunneled service across the Internet. Thus, it will be appreciated that different virtual networks can be formed in different layers using the same underlying physical (or virtual) network resources.

Connections are established between nodes at the edge of the segments where they interface or connect to the shared (Internet) resources which are analogous to the virtual circuits in FIGS. 5, 6, and 7. These may be tunneled connections, or connections built using some other (connection-less) technology.

If we assume T1 and T2 are tunnels, the network databases of IN1.1, IN1.2, IN2.1, and IN2.2 would be augmented with data structures to manage the tunneling protocol at those endpoints, and the links made up from the tunnels. The network database of IN1.1 of FIG. 8 is depicted in FIG. 11 which highlights the "Tunneling Database" and the "IP Database".

Network Database Organization

If we examine the information in the network database for IN1, we will see that it should include configuration and policy information for three separate domains. Furthermore, since the information from the three domains must all coexist in the same physical device, there should be some way to structure the information and control its usage, such that the IN1 device operates correctly in all three domains. If all information for the device IN1 were stored in one monolithic from as is done conventionally, in addition to all the policies for each domain, inter-domain policies would also be required to ensure that information should be is kept private to its own domain.

The illustrative drawing of FIG. 12 is a generalized drawing of a conventional monolithic structure for a database that can be used to implement node IN1 of FIG. 7. The drawing depicts, in a conceptual fashion, an example of the typical organization of information within such a device. The illustrative device includes a first interface attached to VN1.1, a second interface attached to VN2.2 and a third interface attached to the Internet as the shared network resources. To illustrate the complexities in the database design, assume that both the virtual networks being overlaid on the Internet are also (private) IP networks (internets).

Therefore all three networks/domains operate using the IP protocol, each having its own independent IP information to be stored in IN1's network database.

The database includes information such as rules used to articulate and implement administrative policies. The policies as articulated in the information and rules, for example, may include security rules, restrictions on access and dynamic routing protocols. In this illustrative router, the policy information and policy rules used to control the layer 3 IP protocol routing for all three networks are included in a single monolithic database.

However, as explained above, different network domains may have different or perhaps even conflicting policies. In order to provide at least some degree of isolation, additional and complicated "inter-domain" policy mechanisms must be added to manage the conflicts between policies on similar data from different domains. These mechanisms are configured and managed by an administrative authority. The dotted lines in FIG. 12 represent the points at which these inter-domain policy mechanisms would be introduced. The policies would attempt to divide the monolithic network database of node IN1 into three separate domain-specific sections. These dotted lines indicate that separation policy mechanisms are implemented, to provide at least some isolation of the information pertaining to VN1 from the information pertaining to VN2, and also from the information pertaining to the Internet (i.e. shared network resources).

It will be appreciated that the complexity and difficulty in defining and administering the policy mechanisms used to achieve isolation can be great. There is potential for a wide range of policies to be defined between domains. Everything in the spectrum from almost complete openness and sharing of all information between domains, to the other extreme of not sharing anything at all are possible. Certain pieces of a domain's database may want to be kept private (e.g. access

control policy configuration), while other parts are shared to some extent (e.g. summarized routing and addressing information). The types of data, and the extent to which they can all be shared, are all subject to restriction through definition of inter-domain policies.

If we consider each boundary between a pair of domains (i.e. each dotted line through the network database of IN1 in FIG. 12) as a separate policy object, it will also be appreciated that the number of policy objects increases much faster than the number of domains. If D is the number of domains, then P, the number of policy objects can be calculated approximately as:

$$P=(D(D-1))/2$$

Thus, the number of policy objects increases approximately as (a proportion of) the square of the number of domains. In other words, the number of policy objects ordinarily increases much faster than the number of domains, especially as the number of domains gets large.

Another challenge in the administration of virtual networks arises because home or business end station users may wish to change the nature of their connections to the network from time to time. For instance, an end user may wish to utilize a more expensive higher bandwidth connection for business use and a less expensive lower bandwidth connection for home or personal use. Alternatively, for instance, an end user may wish opt to receive a video transmission on a higher bandwidth connection while still receiving other transmissions on lower bandwidth connections. An end user may even wish to change the ISP that he or she uses. Unfortunately, these changes often require intervention by a network administration authority to change the higher level binding between the end user station and the network. More specifically, the binding (or association) between the layer 2/1 virtual circuit service and a layer 3 intermediate device is 'hard', not dynamic, and the higher layer interface generally must be reconfigured by a network administrator to change the binding.

Thus, there has been a need for improved organization of network domain databases and improvements in the ability of a network user to change network domain. The present invention meets these needs.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A is a generalized diagram of a multi-protocol bridge/router.

FIG. 1B is an illustrative example of the topology of and connections.

FIGS. 2A and 2B are network graphs of two illustrative example networks.

FIG. 3 is a network graph of an illustrative network in which the networks of FIGS. 2A and 2B are joined.

FIG. 4 is an illustrative drawings of a segment of a single physical medium capable of carrying multiple information flows which in its own virtual circuit (or channel).

FIG. 5 is an illustrative drawings of two virtual network each made up of two independent segments.

FIG. 6 is an illustrative drawings that provides additional details of some of the physical constituents of the virtual networks of FIG. 5.

FIG. 7 is an illustrative drawings which shows the logical or higher level view of the two virtual network VN1 and VN2 of FIGS. 5 and 6.

FIG. 8 is an illustrative drawings that shows that the Internet can provide the shared network resources of FIGS. 5 and 6.

FIG. 9 is an illustrative drawings that shows tunneling through the Internet to provide the shared resources of FIGS. 5 and 6.

FIG. 10 is a logical or high level view of the two virtual networks of FIG. 9.

FIG. 11 is a generalized illustrative drawing of the organization of node IN1 to achieve tunneling.

FIG. 12 is a conceptual drawing of one possible router configuration that can be used to implement intermediate node IN1 of FIG. 7.

FIG. 13 is a generalized block diagram of a network device that instantiates multiple virtual network machine routers in electronic in accordance with one embodiment of the invention.

FIG. 14 is a generalized block diagram of a network device that instantiates a virtual network machine with multiple layer 2 sub-interface data structures and multiple layer 3 interfaces and binding data structures that associate layer 2 sub-interface data structures and layer 3 interfaces.

FIG. 15 is a generalized block diagram of the network device of FIG. 14, except that one binding data structure has been removed and another binding data structure has been created.

FIG. 16 is a generalized block diagram of a network device that implements a virtual network machine router and a virtual network machine bridge.

FIG. 17 is a generalized block diagram of the network device as in FIG. 16, except that one binding data structure has been removed and another binding data structure has been created.

FIG. 18 is a generalized block diagram of the network device of FIG. 14, except that one binding data structure has been eliminated and another binding data structure has been created.

FIG. 19 is a generalized block diagram of a network device which comprises a computer which instantiates multiple virtual machines in accordance with an embodiment of the invention.

FIG. 20 is generalized block diagram of the network device of FIG. 19 except that one binding data structure has been removed and another binding data structure has been created.

FIG. 21 is a generalized block diagram of a subscriber management system in accordance with a presently preferred embodiment of the invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention comprises a novel apparatus and method for managing operation of network devices that can operate in multiple virtual network domains. The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of particular applications and their requirements. Various modifications to the preferred embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

### Virtual Network Machines

A Virtual Network Machine (VNM) as the term is used herein to describe the collection of processes and mecha-

nisms that operate on a network device to implement the functions of a node in a virtual network. The preferred embodiment for the VNM is as a set of computer programs and related data structures encoded in electronic memory of a network device and used to operate on information, consuming some portion of a network device's computer and memory storage capacity. The functionality of a virtual network machine can be that of a router, bridge or switch, depending on what is configured in its network database. The native resources of a network device include its processor(s), memory, I/O, communication hardware and system software. The native resources of a network device, for example, may include peripheral devices or even a server computer which may, for instance, provide information about end user privileges or virtual network configurations.

Referring to the illustrative drawing of FIG. **13**, there is shown a generalized block diagram of a new structure for the network database of node IN1 from FIGS. **8** and 12 in accordance with one-embodiment of the invention that supports creation of multiple virtual network machines. In this case, the networks device IN1 supports three virtual network machines VNM0, VNM1 and VNM2. In the embodiment of FIG. **13**, assuming again that all three virtual networks operate using the IP protocol, each virtual machine implements the functionality of an IP router, each operating in its own network domain. Each virtual network machine is allocated a portion of the device's native resources. Each virtual network machine runs the IP protocol stack. Each virtual network machine stores its address, policy and control information separately from the others. Thus, each virtual network machine can operate independently of the other virtual network machines, even though it shares native computer resources with the other virtual network machines. This virtual network machine based organization of information therefore provides greater isolation between network domains.

Each virtual machine has its own network database that contains its control information. VNM0 has a network database that causes it to operate as a router that routes information within the Internet network domain. VNM1 has a network database that causes it to operate as a router that routes resource information within network domain VN1. VNM2 has a network database that causes it to operate as a router that routes resource information within network domain VN2.1. *High Speed Networks, TCP/IP and ATM Design Principles*, by William Stallings, Prentice Hall, 1998 provides detailed discussion of router functions and the functions of other network devices.

The VNMs of FIG. **13** may employ multiple different kinds of layer **1** (physical) media to attach to one or more networks. In a presently preferred embodiment, these physical connections include ATM OC-3c/STM1, ATM DS-3/E3, DS-3 Clear Channel, HSSI and 10/100 Base-2 T TX. Resource information is transmitted across these physical connections such as phone lines, DSL or ADSL for example to and from VNM0, VNM1 and VNM2 using layer 2 (data link) protocols. There are layer 2 LAN (local area network) technology and layer 2 WAN (wide area network) technology protocols. Examples of LAN technologies include Ethernet and IEEE 802.3, Fast Ethernet, Token Ring and Fiber Distributed Data Interface. Examples of WAN technologies include Asynchronous Transfer Mode (ATM), Frame Relay, X.25. Point-to-Point (PPP), Integrated Services Digital Network (ISDN) and High-Level Data Link Control (HDLC). Intermediate stations communicate with each other using layer 3 protocols. Layer 3 protocols include Internet Protocol (IP), AppleTalk and Inter Packet Exchange (IPX). Thus, for example, VNM0, VNM1 and VNM2 each employ one or more layer **3** protocols to communicate with other stations of the network(s) to which they are attached.

Thus, the three virtual machines and the different network domains associated with them are isolated from each other in the network device intermediate station of FIG. **13**, and the task of exercising administrative control can be simplified significantly. Since there is no monolithic database that must be maintained to control information transfers across all of the networks to which the three VNMs are attached, the task of administering each database is simplified.

The virtual network machine based organization also simplifies the administration, lowering the cost of operating all three networks. The organization of information along network domain boundaries eliminates the notion of information from two domains residing under a single monolithic structure, and thereby eliminates the need to define inter-domain policies to manage the separation of information within a monolithic database structure. The separation policy mechanisms represented by the dotted lines cutting through the database of FIG. **12** are gone, and a whole set of administrative chores disappears with them. There will be no need to define the complicated inter-domain policies, and no cost associated with administering them. The amount of information that needs to be configured by the administrators is greatly reduced in size and complexity using this method of database organization.

Other benefits can be realized through greater efficiencies in the implementation of such network devices that are possible with this method of network database organization. Further efficiencies are realized through the elimination of the complicated inter-domain policies in virtually all functions of the device. Essentially, each of the virtual machines VNM0, VNM1 and VNM2 operates a separate/independent network device, performing networking functions its own domain.

## Dynamic Binding

The drawing of FIG. **14** shows another illustrative embodiment of the invention. The IP network device of FIG. **14** implements a router that includes three network interfaces NIF3-0, NIF3-1 and NIF3-2. The network device also has a layer 1/2 connection to an Ethernet service. The network device also has a layer 1/2 connection to a virtual circuit service. An Ethernet service sub-interface data structure Eth1 provides the layer 2 Ethernet connection such as sub-interface data structure provides the layer 2 VCC1 connection. For example, the VCC1 sub-interface data structure of FIG. **14** may be kept in a table that identifies all virtual circuit connections, each defining the encapsulation protocol, the packet or cell, data compression technique and the particular layer 2 protocol used on that circuit. The Ethernet sub-interface data structure may include the Ethernet address of the local connection and other parameters to control transmit and receipt of information on the Ethernet segment. A binding data structure B3-0 binds the Ethernet sub-interface data structure to NIF3-0. A binding data structure B3-2 binds the VCC1 sub-interface data structure to NIF3-2. The Ethernet and VCC1 sub-interface data structures are labeled with the prefix "sub" because they are layer 2 constructs which are below the layer 3 interface constructs in the ISO scheme.

Referring to FIG. **14**, binding data structure B3-0 establishes a layer 2/3 connection between the Ethernet sub-interface data structure and NIF3-0, and binding data structure B3-2 establishes a layer 2/3 connection between VCC1

sub-interface data structure and IF3-2. Binding data structure B3-0 causes information transferred across the Ethernet connection to be processed through to NIF3-0. An IP Forwarding/Routing database controls routing of the information out the correct interface. Binding data structure B3-2 causes the information transferred across the VCC1 connection to be processed through NIF3-2.

The VCC1 sub-interface data structure instantiates a virtual circuit connection to the network device of FIG. 14. A virtual circuit connection such as that in FIG. 14 can be created in accord with any of several technologies. A sub-interface data structure like that in FIG. 14 stores the network device's identity of the virtual circuit attached to it. Many virtual circuits can be established across a single physical connection, and many virtual circuits can be connected to a single network device.

FIG. 15 depicts the same intermediate station as in FIG. 14, except the binding B3-0 has been eliminated, and binding B3-1 has been created. Binding B3-1 associates the Ethernet sub-interface data structure Eth-1 with interface NIF3-1. Interface NIF3-2 remains bound to the sub-interface data structure VCC1. The interface NIF3-0 is not bound to any layer 2 construct. It should be noted that an unbound interface construct generally would represent a misconfiguration in a typical earlier intermediate station.

FIG. 16 depicts yet another illustrative embodiment of the invention. The network device of FIG. 16 implements an IP router function and a bridging function. The router includes two interfaces NIF4-1 and NIF4-2. The bridge includes a bridge interface BR4-0. A network database that implements the bridge function includes a list of network stations reachable through each of the bridge's interfaces. The network device also has a layer 1/2 connection to an Ethernet service. The network device also has a layer 1/2 connection to a virtual circuit service VCC1. An Ethernet service sub-interface data structure Eth1 provides information concerning the Ethernet connection such as a VCC1 sub-interface data structure provides information concerning the VCC1 connection. A binding data structure B4-0 binds the Ethernet sub-interface data structure to NIF4-0. A binding data structure B4-2 binds the VCC1 sub-interface data structure to NIF4-2. NIF4-1 is unbound.

FIG. 17 depicts the same network device as in FIG. 16, except the binding B4-0 has been eliminated, and binding B4-1 has been created. Binding B4-1 associates the Ethernet sub-interface data structure with interface NIF4-1 of virtual router VM4. Interface NIF4-2 remains bound to the sub-interface data structure VCC1. The interface BR4-0 is not bound to any layer 2 construct. These changes in binding effectively redefines the service available on the Ethernet segment from a bridged or layer 2 service, to a routed or layer 3 service. In a presently preferred embodiment of the invention, these bindings can be changed without reconfiguration of any other interface construct or circuit construct. In a typical earlier intermediate station, the bindings between the higher and lower layers are implicit, and a change in the implicit bindings applied to the bridge and router interface constructs typically would have required a modification of these interface constructs. A present embodiment of the invention does not require such modification.

FIG. 18 depicts the same network device as in FIG. 14, except the binding B3-0 has been eliminated and binding B3-2A has been created. Binding B3-2A associates the Ethernet sub-interface data structure with the NIF3-2 interface. Binding B4-2 associates the VCC1 sub-interface data structure with NIF3-2. Interfaces NIF3-0 and NIF3-1 are unbound. This change in bindings causes both the Ethernet and the virtual circuit lower layer services to be associated with a single higher layer IP construct, NIF3-2.

FIG. 19 shows a network device which comprises a computer which instantiates multiple virtual network machines VNM5 and VNM6. VNM5 implements IP router functionality. It includes network interfaces NIF5-0 and NIF5-1. VNM6 also implements IP router functionality. It includes two interfaces NIF6-0 and NIF6-1. The network device of FIG. 19 has two layer 1/2 connections to a virtual circuit service. Sub-interface data structure VCC1 instantiates one of the connections to the device. Sub-interface VCC2 instantiates the other connection to the device. A binding data structure B5-0 binds the VCC1 sub-interface data structure to NIF5-0 of VNM5. A binding data structure B6-2 binds the VCC2 sub-interface data structure to interface NIF6-1 of VNM6. VNM5 and VNM6 each use the IP protocol suite to communicate with other stations of the network(s) to which they are attached.

FIG. 20 depicts the same network device as in FIG. 19, except the binding B5-0 has been eliminated and binding B6-0 has been created. The binding B6-0 data structure associates VCC1 sub-interface data structure with NIF6-0 of VNM6. Binding data structure B6-1 binds sub-interface data structure VCC2 to NIF6-1. Neither of the VNM5 interfaces NIF5-0 and NIF5-1 are bound.

In FIGS. 14 to 20, bindings are shown as data structures connected to other data structures by line segments. In one preferred embodiment, the line segments each represent a pair of bi-directional pointers; the first pointer points from the binding to the higher or lower layer data structures and the second is opposite the first, pointing from the higher or lower layer data structure to the binding data structure. Alternatively, the binding could be implemented as indices or identifiers in a table, for example. Dynamic binding is accomplished by creating and or deleting binding data structures-and or changing the values of the pointers or indices so they operate on different data structures. It will be appreciated that actual changing of the bindings can be accomplished through entries in a command line interface to the network device or automataically by snooping the information flow through the device, for example.

The illustrative drawing of FIG. 21 is a generalized block diagram of a subscriber management system in accordance with a presently preferred embodiment of the invention. A subscriber is a user of network services. The system includes a computer with layer 1/2 connections to subscriber end stations and with layer 1/2 connections to network devices that provide access to other networks.

The system can form a multiplicity of layer 1/2 subscriber end station connections. In a present embodiment, the layer 1/2 connections to subscriber end stations include virtual circuit connections. The system memory stores a multiplicity of sub-interface data structures that instantiate the multiplicity of virtual circuit connections through which subscriber end stations communicate with the subscriber management system.

The system instantiates in memory a plurality of virtual network machines. Each VNM of the embodiment of FIG. 21 implements the functionality of a router. There are nine illustrative VNM routers shown in FIG. 21 labeled VNMr1–VNMr-9. Each VNM router includes interfaces in its database. Each VNM router runs at least one layer 3 protocol suite. Each VNM router may run one or more adaptive routing algorithms. The interfaces of each VNM router provide access to a network that is isolated from the

networks accessed through the interfaces of the other VNM routers. For example, the interface to VNMr-4 provides layer 3 access to the network that includes ISP#2. The interface to VNMr-5 provides layer 3 access to the network that includes Corporate-Private-Network#A. The interface to VNMr-6 provides layer 3 access to the network that includes ISP#4. The networks with ISP#2, Corporate-Private-Network#A and ISP#4 are isolated from each other. The databases associated with VNMr-4, VNMr-5 and VNMr-6 to control access to networks across these respective interfaces. Each of these three VNM databases can be administered separately.

In operation a subscriber might establish a point-to-point connection with the subscriber management system. A server that runs software that runs authentication, authorization and accounting protocols (AAA) searches for a record that identifies the user. Authentication is the process of identifying and verifying a user. For instance, a user might be identified by a combination of a username and a password or through a unique key. Authorization determines what a user can do after being authenticated, such as gaining access to certain end stations information resources. Accounting is recording user activity. In the present embodiment, AAA involves client software that runs on the subscriber management system and related access control software that runs either locally or on a remote server station attached to the network. The present embodiment employs Remote Authentication Dial-In User Service (RADIUS) to communicate with a remote server. An example of an alternative AAA protocol is Terminal Access Controller Access Control System (TACACS+). RADIUS and TACAS+ are protocols that provide communication between the AAA client on a router and access control server software.

The subscriber record includes information concerning the network to which the subscriber's virtual circuit connection should be bound. Typically, the subscriber will employ a PVC. Based upon the information in the subscriber record, a binding data structure, like that described in reference to FIGS. **14** to **20**, will be created to associate the sub-interface data structure that instantiates the PVC in the subscriber management system memory with the interface to the VNM router that accesses the network identified for the subscriber in the subscriber record.

Moreover, the subscriber record may provide multiple possible binding options for the subscriber. For instance, the subscriber may specify the creation of a binding that is which is to be employed during business hours and which binds the subscriber to VNMr-5 which provides layer 3 network access to the Corporation-Private-Network#. The same record may specify another binding which is to be employed only during non-business hours and which binds to VNM#4 which provides layer 3 network access to ISP#2. Thus, the bindings can be changed. They are dynamic.

Various modifications to the preferred embodiments can be made without departing from the spirit and scope of the invention. Thus, the foregoing description is not intended to limit the invention which is described in the appended claims in which:

What is claimed is:

**1**. A computer implemented method comprising:

routing Internet Protocol (IP) packets within a first Internet Service Provider's (ISP's) domain from a single network device with a first database, the first database including addresses of the first ISP's domain; and

routing IP packets within a second ISPs domain from the single network device with a second database, the second database being separate from the first database and including addresses of the second ISP's domain.

**2**. The computer implemented method of claim **1**, wherein the first database also includes control and policy information for the first ISP's domain and the second database includes control and policy information for the second ISP's domain.

**3**. The computer implemented method of claim **1** further comprising connecting a subscriber to the first ISP's domain with an authentication, authorization and accounting protocol.

**4**. The computer implemented method of claim **1** further comprising:

routing IP packets within a corporation's domain from the single network device with a third database, the third database being separate from the first and second databases, wherein said third database includes addresses of the corporation's domain.

**5**. The computer implemented method of claim **1** further comprising:

providing the corporation administrative control of the third database, but not the first and second databases;

providing the first ISP administrative control of the first database, but not the second and third databases; and

providing the second ISP administrative control of the second database, but not the first and third databases.

**6**. The method of claim **1** further comprising routing the packets within the first ISP's domain with a global database that includes globally known addresses if the packets cannot be routed within the first ISP's domain with the first database.

**7**. A memory having a set of one or more programs stored thereon to cause a single network device to perform operations comprising:

maintaining a first database separately from a second database in the single network device, the first database having addresses for a first Internet Service Provider's (ISP's) domain and the second database having addresses for a second ISP's domain;

routing Internet Protocol (IP) packets within the first ISP's domain from the single network device with the first database; and

routing IP packets within the second ISP's domain from the single network device with the second database.

**8**. The memory of claim **7** further comprising providing access to a subscriber to the first ISP's domain with an authentication, authorization, and accounting protocol.

**9**. The computer implemented method of claim **7** further comprising:

maintaining a third database separately from the first and second databases, wherein the third database has addresses for a corporation's domain; and

routing IP packets within the corporation's domain with the third database from the single network device.

**10**. The computer implemented method of claim **9** further comprising:

providing the first ISP administrative control of the first database, but not the second or third databases;

providing the second ISP administrative control of the second database, but not the first or third databases; and

providing the corporation administrative control of the third database, but not the first or second databases.

**11**. The memory of claim **7** wherein the set of one or more programs cause the single network device to perform operations further comprising:

maintaining a third database separately from the first and second databases, wherein the third database has addresses of a backbone; and

routing IP packets within the first ISP's domain with the third database if they cannot be routed with the first database.

12. A single network device comprising:

an electronic memory having

a first database of network addresses of a first network domain that is administered by a first Internet Service Provider (ISP);

a second database of network addresses of a second network domain that is administered by a second ISP, the second database being isolated from the first database; and

a set of one or more processors to execute a set of instructions that cause the single network device to route a first set of packets of the first network domain with the first database and to route a second set of packets of the second network domain with the second database.

13. The single network device of claim 12 wherein the packets are IP packets.

14. The single network device of claim 12 wherein the packets are layer 2 packets.

15. The single network device of claim 12 wherein the first set of packets are transmitted from a subscriber of the first ISP and the second set of packets are transmitted from a subscriber of the second ISP.

16. The single network device of claim 12 further comprising:

the electronic memory further having

a third database to store network addresses of a third network domain that is administered by a corporation, the third database being isolated from the first and second databases; and

the set of processors to execute the set of installations to further cause the single network device to route a third set of packets of the third network domain with the third database.

17. The single network device of claim 12 further comprising:

the electronic memory having a third database of addresses of a network provider that is administered by the network provider, the third database being isolated from the first and second databases; and

the set of processors to execute the set of instructions that further cause the single network device to route the first set of packets with the third database if they cannot be routed with the first database.

18. A method comprising:

routing packets for a first set of subscribers with a fist virtual router, and routing packets for a second set of subscribers with a second virtual router, the first and second virtual routers being isolated from each other within a single network device, the first set of subscribers subscribing to a first Internet Service Provider (ISP) and the second set of subscribers subscribing to a second ISP;

providing administrative control of the first virtual router, which includes a first network database, used by the first virtual router to route packets, of network device addresses within the first ISP's domain and control and policy information for the first ISP's domain, to the first ISP; and

providing administrative control of the second virtual router, which includes a second network database, used by the second virtual router to route packets, of network device addresses within the second ISP's domain and control and policy information for the second ISP's domain, to the second ISP, wherein the first ISP does not have administrative control of the second network database and the second ISP does not have administrative control of the first network database.

19. The method of claim 18 wherein the packets are layer 2 packets.

20. The method of claim 18 wherein providing administrative control of the first virtual router comprises allowing the first ISP to modify the first network address database and the control and policy information governing the first ISP's domain.

21. The method of claim 18 further comprising:

providing a network provider administrative control of a global virtual router including a global network database in the single network device.

22. The method of claim 18 further comprising:

routing packets for a third set of subscribers with a third virtual router; and

providing administrative control of the third virtual router, which includes a third network database of network device addresses within a corporation's domain and control and policy information for the corporation's domain, wherein the corporation has administrative control of the third virtual router but not the first and second virtual routers.

23. The method of claim 22 further comprising providing a network provider access to the first, second and third network databases and a global network database, said global network database being in said single network device.

24. The method of claim 18 wherein the packets are layer 3 packet.

25. The method of claim 18 further comprising:

connecting the first and second set of subscribers to the single network device in accordance with an a authorization, authentication and accosting protocol.

26. An electronic memory encoded with a set of instructions, which when executed on a single network device, cause said single network device to perform operations comprising:

creating a plurality of collections of processes and mechanisms for implementing router functionality, each of the plurality of collections of processes and mechanisms operating on a different network database including addresses and control and policy information;

separately storing the network database of each of the plurality of collections of processes and mechanisms; and

each of the plurality of collections of processes and mechanisms routing packets within a different administrative domain with its network database and in accordance with its control and policy information.

27. The electronic memory of claim 26 wherein the packets are layer 2 packets.

28. The electronic memory of claim 26 wherein each of the plurality of collections of processes and mechanisms runs its own IP stack.

29. The electronic memory of claim 26 wherein at least one of the different administrative domains is administered by an Internet Service Provider.

30. The electronic memory of claim 26 wherein at least one of the different administrative domains is administered by a corporation.

31. A single network device comprising:

plurality of virtual network machines that are individually isolated, each of the plurality of virtual network machines to route packets within a different administrative domain with

a network address database for the different administrative domain, and

control and policy information for the different administrative domain;

a first port to transmit and receive said packets to and from subscribers; and

a second port to transmit and receive said packets to and from the Internet.

32. The single network device of claim 31 wherein the plurality of virtual network machines are virtual IP routers.

33. The single network device of claim 31 wherein the different administrative domains include an Internet Service Provider domain and a corporate domain.

34. The single network device of claim 31 further comprising a third port to transmit and receive a second set of packets to and from a corporate network domain.

35. A single network device comprising:

communication hardware;

a set of one or more processors coupled with the communication hardware; and

an electronic memory coupled with the communication hardware and the set of processors, the electronic memory encoded with a set of instructions to cause the set of processors to,

host a first virtual router that includes a first network database of network device addresses within a first Internet Service Provider's (ISP's) domain and control and policy information for the first ISP's domain, and

host a second virtual router, isolated from the first virtual router, that includes a second network database of network device addresses within a second ISP's domain and control and policy information for the second ISP's domain,

route packets for a first set of subscribers with the communication hardware and the first virtual router, wherein the first set of subscribers subscribe to the first ISP,

route packets for a second set of subscribers with the communication hardware and the second virtual router, wherein the second set of subscribers subscribe to the second ISP,

provide the first ISP administrative control of the first virtual router but not the second virtual router, and

provide the second ISP administrative control of the second virtual router, but not the first virtual router.

36. The single network device of claim 35 further comprising the electronic memory to host a global network database.

37. The single network device of claim 35 further comprising:

the electronic memory to host a third virtual router that includes a third network database of network device addresses within a corporation's domain and control and policy information for the corporation's domain, and the set of instructions to further cause the set of processors to,

route packets for a third set of subscribers with the communication hardware and the third virtual router, and

provide the corporation administrative control of the third virtual router, but not the first and second virtual routers.

38. The single network device of claim 35, wherein the set of instructions further cause the set of processors to switch packets for a third set of subscribers with the communication hardware and the first virtual router.

39. The single network device of claim 35 wherein the packets are layer 3 packets.

40. A network comprising:

a set of one or more networks;

a set of one or more end stations communicating, for a first set of one or more subscribers of a first Internet Service Provider (ISP) and for a second set of one or more subscribers of a second ISP, packets;

a single network access device coupled between the set of networks and the set of end stations, the single network access device having,

communication hardware;

an electronic memory coupled with the communication hardware, the electronic memory having stored therein,

a first network database, controllable for administration by the first ISP but not the second ISP, including network device addresses and control and policy information for the first ISP,

a second network database, controllable for administration by the second ISP but not the first ISP, including network device addresses and control and policy information for the second ISP, wherein the fast network database and the second network database are isolated from each other; and

a set of one or more processors, coupled with the communication hardware and the electronic memory, routing said packets being communicated for the first set of subscribers with the communication hardware and a first virtual router that includes the first network database, and routing said packets being communicated for the second set of subscribers with the communication hardware and a second virtual router that includes the second network database.

41. The network of claim 40 further comprising the electronic memory having stored therein a global network database.

42. The network of claim 40 fiber comprising:

the set of one or more networks including a virtual network of a corporation;

the set of end stations communicating, for a third set of one or more subscribers of the corporation, packets;

the electronic memory having stored therein a third network database, controllable for administration by the corporation but not the first ISP nor the second ISP, including network device addresses and control and policy information for the corporation; and

the set of processors routing said packets being communicated for the third set of subscribers with the communication hardware and a third virtual router that includes the third network database.

43. The network of claim 40 further comprising the set of processors switching packets for a third set of one or more subscribers with the communication hardware and the first virtual router.

44. The network of claim 40 wherein the packets are layer 3 packets.

45. An electronic memory encoded with a set of instructions, which when executed on a single network device, cause said single network device to perform operations comprising:

routing packets for a first set of subscribers with a first virtual router, and routing packets for a second set of subscribers with a second virtual router, the first and second virtual routers being isolated from each other within the single network device, the first set of subscribers subscribing to a first Internet Service Provider (ISP) and the second set of subscribers subscribing to a second ISP;

providing administrative control of the first virtual router, which includes a first network database of network device addresses within the first ISP's domain and control and policy information for the first ISP's domain, to the first ISP; and

providing administrative control of the second virtual router, which includes a second network database of network device addresses within the second ISP's domain and control and policy information for the second ISP's domain, to the second ISP, wherein the first ISP does not have administrative control of the second network database and the second ISP does not have administrative control of the first network database.

**46**. The electronic memory of claim **45**, wherein the operations further comprise:

routing packets for the first and second set of subscribers with a third virtual router, the third virtual routing including a global network database.

**47**. The electronic memory of claim **45** wherein the operations further comprise:

routing packets for a third set of subscribers with a third virtual router, the third virtual router being isolated from the first and second virtual router within the single network device; and

providing a corporation administrative control of the third virtual router, which includes a third network database of network device addresses of the corporation and control and policy information for the corporation, wherein the corporation has administrative control of the third virtual router but not the first and second virtual routers.

**48**. The electronic memory of claim **45**, wherein the operations further comprise:

switching packets for a third set of one or more subscribers with the first virtual router.

**49**. A method in a single network device comprising:

creating a plurality of collections of processes and mechanisms for implementing router functionality, each of the plurality of collections of processes and mechanisms operating on a different network database including addresses and control and policy information;

separately storing the network database of each of the plurality of collections of processes and mechanisms; and

each of the plurality of collections of processes and mechanisms routing packets within a different administrative domain with its network database and in accordance with its control and policy information.

**50**. The computer implemented method of claim **49** wherein at least one of the plurality of collections of processes and mechanisms switching packets within its administrative domain with its network database.

**51**. The computer implemented method of claim **49** wherein each of the plurality of collections of processes and mechanisms runs its own IP stack.

**52**. The computer implemented method of claim **49** wherein at least one of the different administrative domains is administered by an Internet Service Provider.

**53**. The computer implemented method of claim **49** wherein at least one of the different administrative domains is administered by a corporation.

**54**. The computer implemented method of claim **49** wherein the different network database of each of the plurality of collections of processes and mechanisms includes addressing information, control information, and policy information.

**55**. A single network device comprising:

a set of one or more processors; and

an electronic memory coupled with the set of processors, the electronic memory

having a set of instructions to cause the set of processors to,

create a plurality of collections of processes and mechanisms, each of the plurality of collections of processes and mechanisms to operate on a different network database including control and policy information, and to route packets within a different administrative domain with its network database and in accordance with its control and policy information, and

separately store the different network database of each of the plurality of collections of processes and mechanisms.

**56**. The single network device of claim **55** further comprising at least one of the collections of processes and mechanisms to switch packets with its network database and in accordance with its control and policy information.

**57**. The single network device of claim **55** wherein the set of instructions further cause the set of processors to independently run an Internet Protocol stack for each of the plurality of collections of processes and mechanisms.

**58**. The single network device of claim **55** wherein at least one of the different administrative domains is administered by an Internet Service Provider.

**59**. The single network device of claim **55** wherein at least one of the different administrative domains is administered by a corporation.

**60**. The single network device of claim **55** wherein the different network database of each of the plurality of collections of processes and mechanisms is to include addressing information, control information, and policy information.

**61**. A network comprising:

a set of one or more networks;

a set of one or more end stations communicating packets with the set of networks; and

a single network device coupled between the set of networks and the set of end stations, the single network device having a plurality of collections of processes and mechanisms, each of the plurality of collections of processes and mechanisms,

operating on a different network database including addresses and control and policy information, wherein the network database operated on by each of the collection of processes and mechanisms is stored separately, and

routing packets within a different administrative domain with its network database and in accordance with its control and policy information.

**62**. The network of claim **61** further comprising at least one of the plurality of collections of processes and mechanisms switching packets with its network database and in accordance with its control and policy information.

**63**. The network of claim **61** each of the plurality or collections of processes and mechanisms independently running an IP stack.

64. The network of claim 61 wherein at least one of the different administrative domains is administered by an Internet Service Provider.

65. The network of claim 61 wherein at least one of the different administrative domains is administered by a corporation.

66. A network comprising:

a set or one or more networks;

a set of one or more end stations communicating packets with the set of networks, and

a single network device coupled between the set of networks and the set of end stations, the single network device having,

a first virtual network machine transmitting certain of said packets for a first subscriber in accordance with a first network database of a first administrative domain, the first database having addressing and policy information of the first administrative domain, and

a second virtual network machine, which is isolated from the first virtual network machine, transmitting certain packets for a second subscriber in accordance with a second network database, the second network database having addressing and policy information for a second administrative domain.

67. The network of claim 66 wherein the first administrative domain is administered by a corporation.

68. The network of claim 66 wherein the first administrative domain is administered by an Internet Service Provider.

69. The network of claim 66 wherein the first administrative domain provides a first service and the second administrative domain provides it second service.

70. The network of claim 66 wherein the addressing information in the first database includes layer 2 addressing information.

71. The network of claim 66 wherein the addressing information in the first database include layer 3 addressing information.

72. A single network device comprising:

a first set of one or more ports to receive IP packets from a first and second set of one or more subscribers;

a second set of one or more ports to transmit IP packets over a first network domain;

a machine-readable medium having stored therein a set of instructions to cause the single network device to,

instantiate a first and second virtual router, which are virtually-independent but share a set of physical resources within the single network device,

the first virtual router to route within a second network domain, which is layered upon the first network domain, WP packets from the first set of subscribers using a first network database that includes IP addresses, control and policy information defined for the second network domain, and

the second virtual router to route within a third network domain, which is layered upon the first network domain and shares the first network domain's physical resources with the second network domain, IP packets from the second set of subscribers using a second network database that includes IP addresses and control and policy information defined for the third network domain,

maintain separation between the first and second network databases so as to avoid management of inter-domain policies, wherein avoidance of inter-domain policies eases administrative tasks,

provide for independent administration of the first and second network databases, wherein independent administration maintains administrative integrity of the first and second network databases.

73. The single network device of claim 72 wherein the first set of subscribers are subscribers of a corporate network and the second set of subscribers are subscribers of an Internet service provider.

74. The single network device of claim 72 wherein the first network domain is a layer 3 network domain and the second and third network domains are layer 3 network domains.

75. The single network device of claim 72 further comprising the set of instructions to cause the single network device to tunnel IP packets of the first set of subscribers between separate physical locations of a virtual private network with the first virtual router.

76. The single network device of claim 72 further comprising the set of instructions to cause the single network device to process the first and second set of subscribers in accordance with an authorization, authentication and accounting protocol.

77. A single network device comprising:

a first set of one or more ports to receive IP packets from subscribers;

a second set of one or more ports to transmit IP packets over a first network domain;

a machine-readable medium having stored therein a set of instructions to cause the single network device to,

instantiate different virtual routers for different network domains, which are layered upon the first network domain, using separate unshared inter-domain policy free, independently administrable network databases, wherein each of the separate unshared inter-domain policy free, independently administrable network databases includes IP addresses, control and policy information defined for its one of the different network domains, and

route IP packets of different ones of the subscribers using those of the virtual routers for the different ones of the network domains to which those subscribers currently belong.

78. The single network device of claim 77 wherein the subscribers includes subscribers of a corporate network and subscribers of different Internet service providers.

79. The single network device of claim 77 wherein the first network domain is a layer 3 network domain and the different network domains layered upon the first network domain are layer 3 network domains.

80. The single network device of claim 77 further comprising the set of instructions to cause the single network device to tunnel IP packets of certain of the subscribers between separate physical locations of a virtual private network with the one of the different virtual routers to which the certain of the subscribers belong.

81. The single network device of claim 77 further comprising the set of instructions to cause the single network device to process the subscribers in accordance with an authorization, authentication and accounting protocol.

* * * * *

US006230271B1

(12) **United States Patent**
**Wadlow et al.**

(10) **Patent No.: US 6,230,271 B1**
(45) **Date of Patent: May 8, 2001**

(54) **DYNAMIC POLICY-BASED APPARATUS FOR WIDE-RANGE CONFIGURABLE NETWORK SERVICE AUTHENTICATION AND ACCESS CONTROL USING A FIXED-PATH HARDWARE CONFIGURATION**

(75) Inventors: **Thomas A. Wadlow**, San Francisco; **Joseph P. Kevin**, Hayward, both of CA (US)

(73) Assignee: **Pilot Network Services, Inc.**, Alameda, CA (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/009,923**

(22) Filed: **Jan. 20, 1998**

(51) **Int. Cl.**[7] ..... **H04L 9/00**
(52) **U.S. Cl.** ..... **713/201**; 713/200
(58) **Field of Search** ..... 713/200, 201, 713/202; 709/218, 227, 250; 714/38

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | |
|---|---|---|---|---|
| 5,265,221 | * | 11/1993 | Miller | 395/725 |
| 5,515,376 | * | 5/1996 | Murthy et al. | 370/85.13 |
| 5,577,209 | * | 11/1996 | Boyle et al. | 395/200.06 |
| 5,606,668 | * | 2/1997 | Shwed | 713/201 |
| 5,610,905 | * | 3/1997 | Murthy et al. | 370/401 |
| 5,623,601 | * | 4/1997 | Vu | 395/187.01 |
| 5,781,550 | * | 7/1998 | Templin et al. | 370/401 |
| 5,787,253 | * | 7/1998 | McCreery et al. | 395/200.61 |
| 5,832,211 | * | 11/1998 | Blakley, III et al. | 395/188.01 |
| 5,835,726 | * | 11/1998 | Shwed et al. | 709/229 |
| 5,864,683 | * | 1/1999 | Boebert et al. | 395/200.79 |
| 5,884,024 | * | 3/1999 | Lim et al. | 395/187.01 |

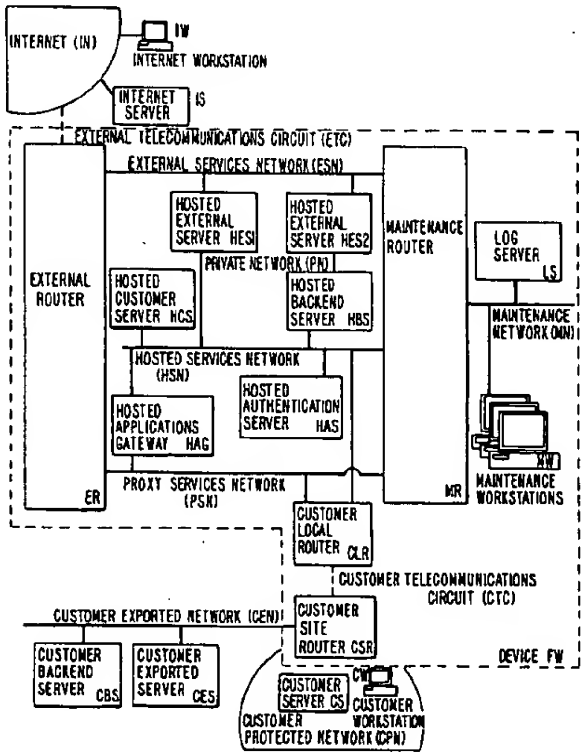
* cited by examiner

*Primary Examiner*—Dieu-Minh T. Le
(74) *Attorney, Agent, or Firm*—Townsend and Townsend and Crew LLP; Philip H. Albert

(57) **ABSTRACT**

A secure interface between a private network and a public network are disclosed. The interface includes a collection of routing devices, telecommunications devices, packet-filtering devices, applications-filtering devices, monitoring and maintenance devices organized to enforce a wide variety of desired customer security policies. The apparatus allows a central service provider to install and maintain a collection of similar apparatus and support a number of customers with widely varying security policies and allows a rapid change in the security policy of a given customer, without affecting the security of other customers, and without requiring the customer to understand or modify the underlying apparatus.

**7 Claims, 17 Drawing Sheets**



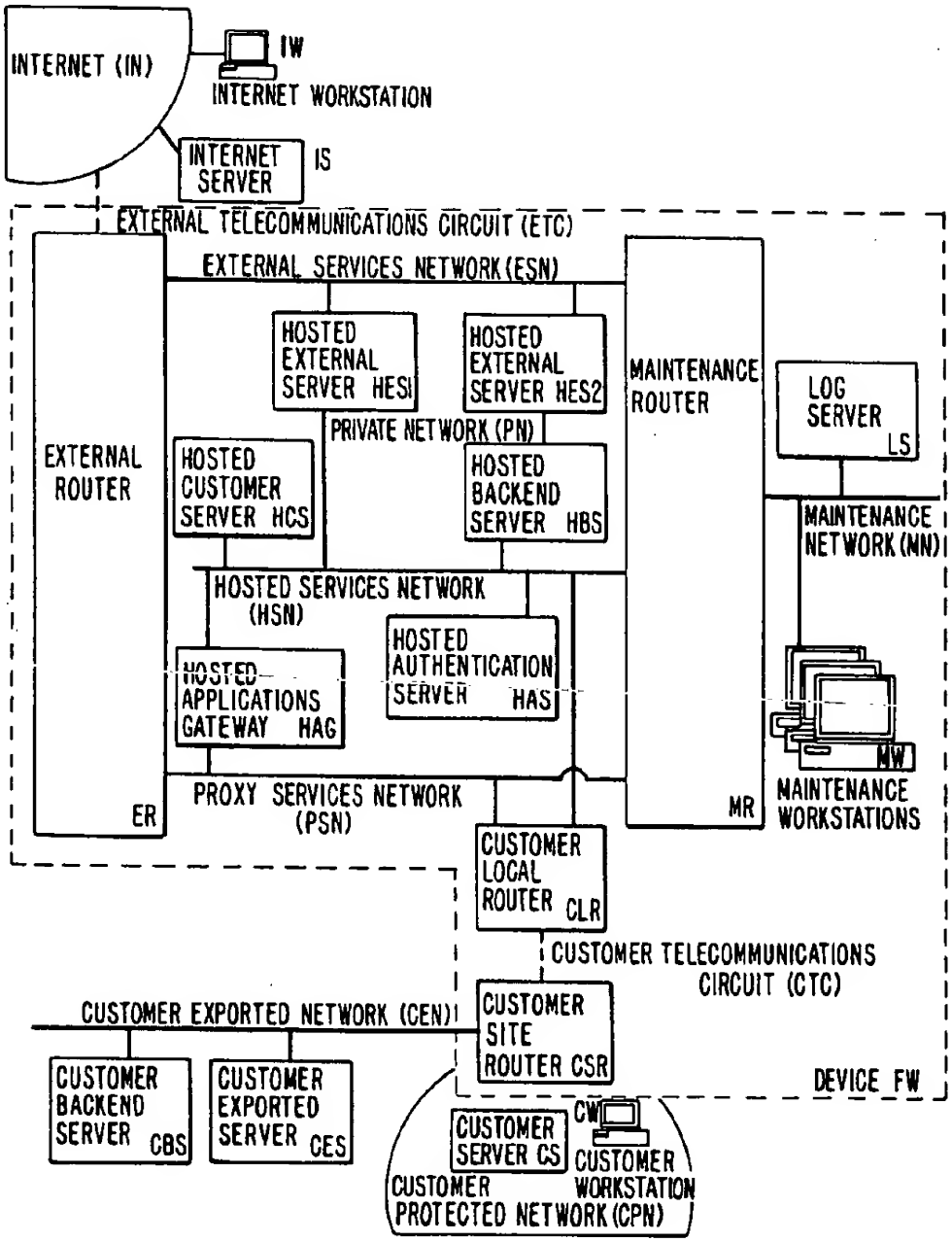

INTERNET (IN)
IW
INTERNET WORKSTATION
INTERNET SERVER
IS
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
EXTERNAL SERVICES NETWORK (ESN)
HOSTED EXTERNAL SERVER HES1
HOSTED EXTERNAL SERVER HES2
MAINTENANCE ROUTER
LOG SERVER LS
EXTERNAL ROUTER
PRIVATE NETWORK (PN)
HOSTED CUSTOMER SERVER HCS
HOSTED BACKEND SERVER HBS
MAINTENANCE NETWORK (MN)
HOSTED SERVICES NETWORK (HSN)
HOSTED AUTHENTICATION SERVER HAS
HOSTED APPLICATIONS GATEWAY HAG
MW
MAINTENANCE WORKSTATIONS
PROXY SERVICES NETWORK (PSN)
ER
MR
CUSTOMER LOCAL ROUTER CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
CUSTOMER EXPORTED NETWORK (CEN)
CUSTOMER SITE ROUTER CSR
CUSTOMER BACKEND SERVER CBS
CUSTOMER EXPORTED SERVER CES
DEVICE FW
CUSTOMER SERVER CS
CW
CUSTOMER WORKSTATION
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 1.

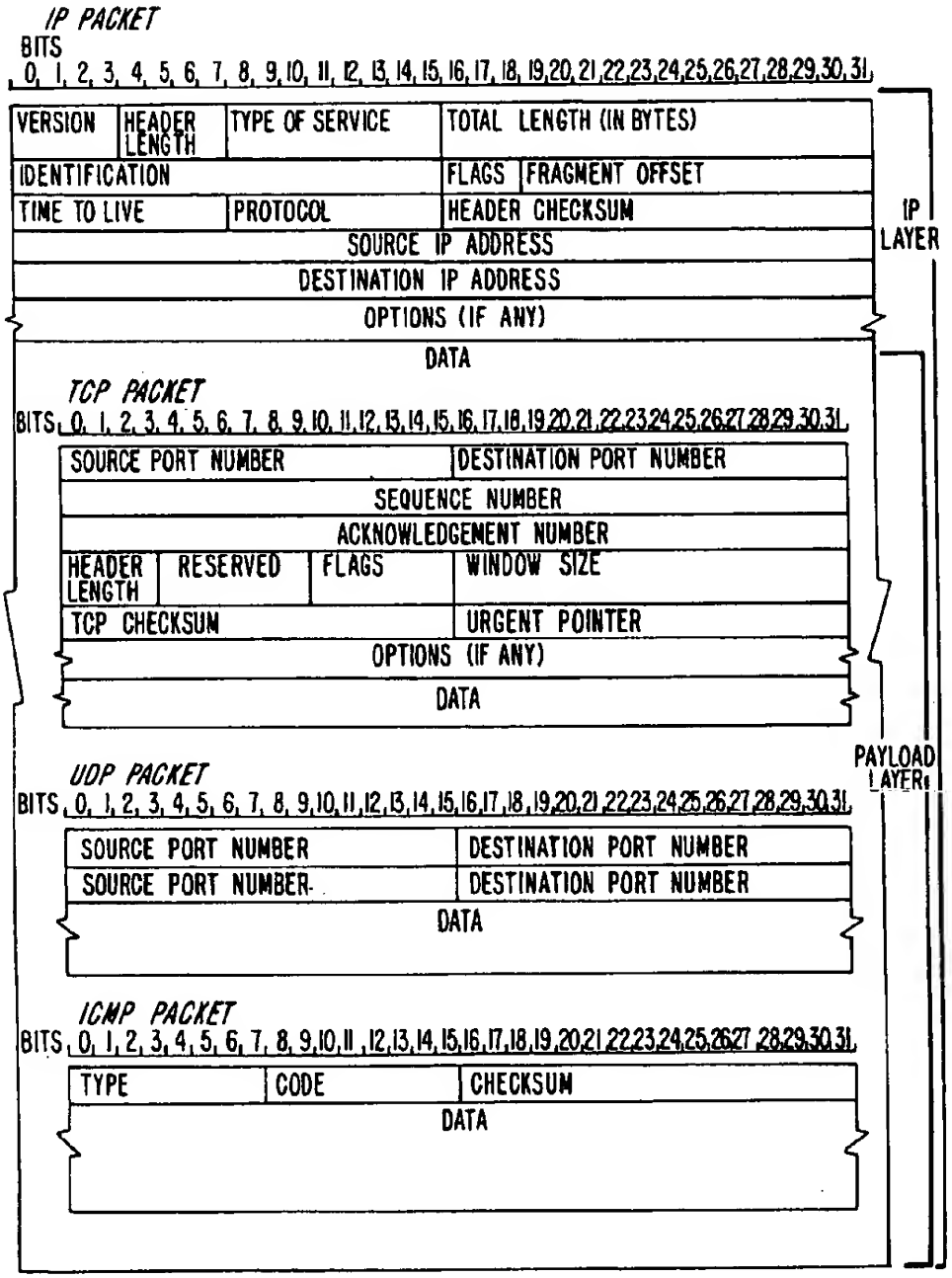



FIG. 2. (PRIOR ART)

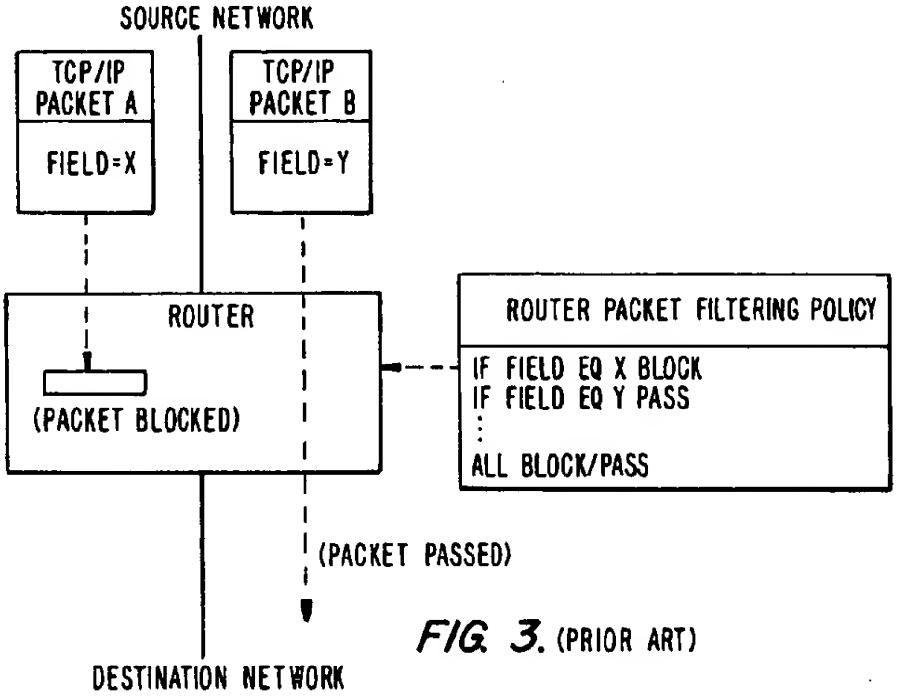



FIG. 3. (PRIOR ART)

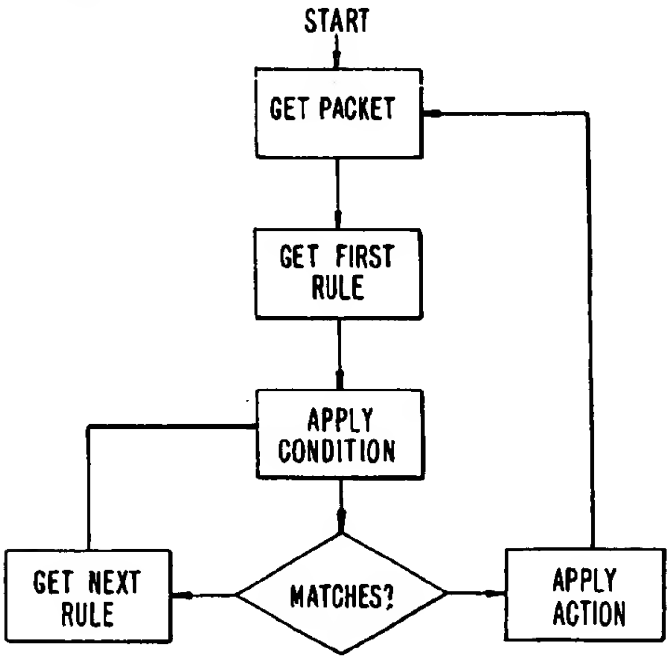



FIG. 3A. (PRIOR ART)

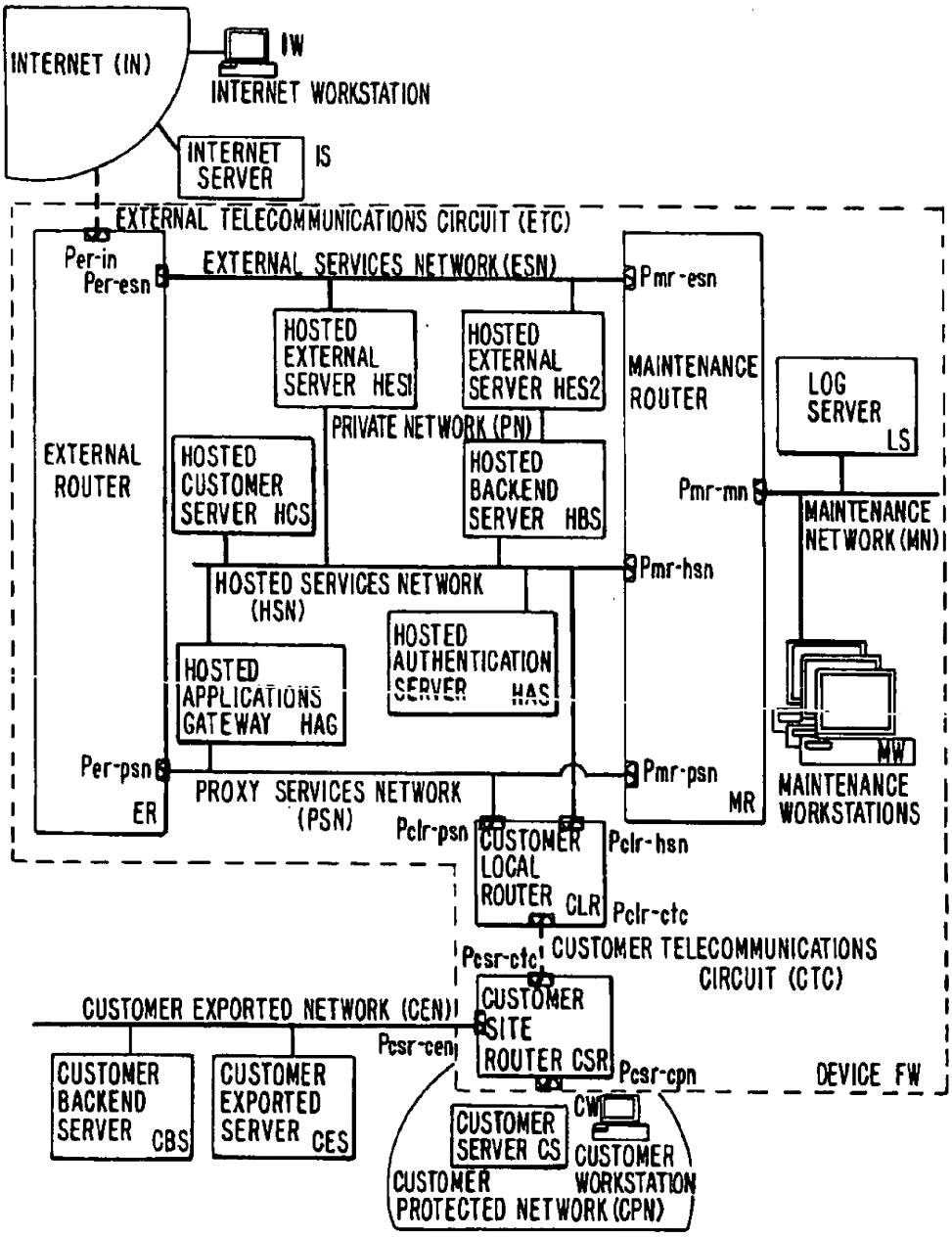

INTERNET (IN)
IW
INTERNET WORKSTATION
INTERNET SERVER
IS
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
Per-in
Per-esn
EXTERNAL SERVICES NETWORK (ESN)
Pmr-esn
HOSTED EXTERNAL SERVER HES1
HOSTED EXTERNAL SERVER HES2
MAINTENANCE ROUTER
LOG SERVER LS
PRIVATE NETWORK (PN)
EXTERNAL ROUTER
HOSTED CUSTOMER SERVER HCS
HOSTED BACKEND SERVER HBS
Pmr-mn
MAINTENANCE NETWORK (MN)
HOSTED SERVICES NETWORK (HSN)
Pmr-hsn
HOSTED AUTHENTICATION SERVER HAS
HOSTED APPLICATIONS GATEWAY HAG
MW
Per-psn
ER
PROXY SERVICES NETWORK (PSN)
Pmr-psn
MR
MAINTENANCE WORKSTATIONS
Pclr-psn
CUSTOMER LOCAL ROUTER CLR
Pclr-hsn
Pclr-ctc
Pcsr-ctc
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
CUSTOMER EXPORTED NETWORK (CEN)
CUSTOMER SITE ROUTER CSR
Pcsr-cen
Pcsr-cpn
DEVICE FW
CUSTOMER BACKEND SERVER CBS
CUSTOMER EXPORTED SERVER CES
CW
CUSTOMER SERVER CS
CUSTOMER WORKSTATION
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 4.

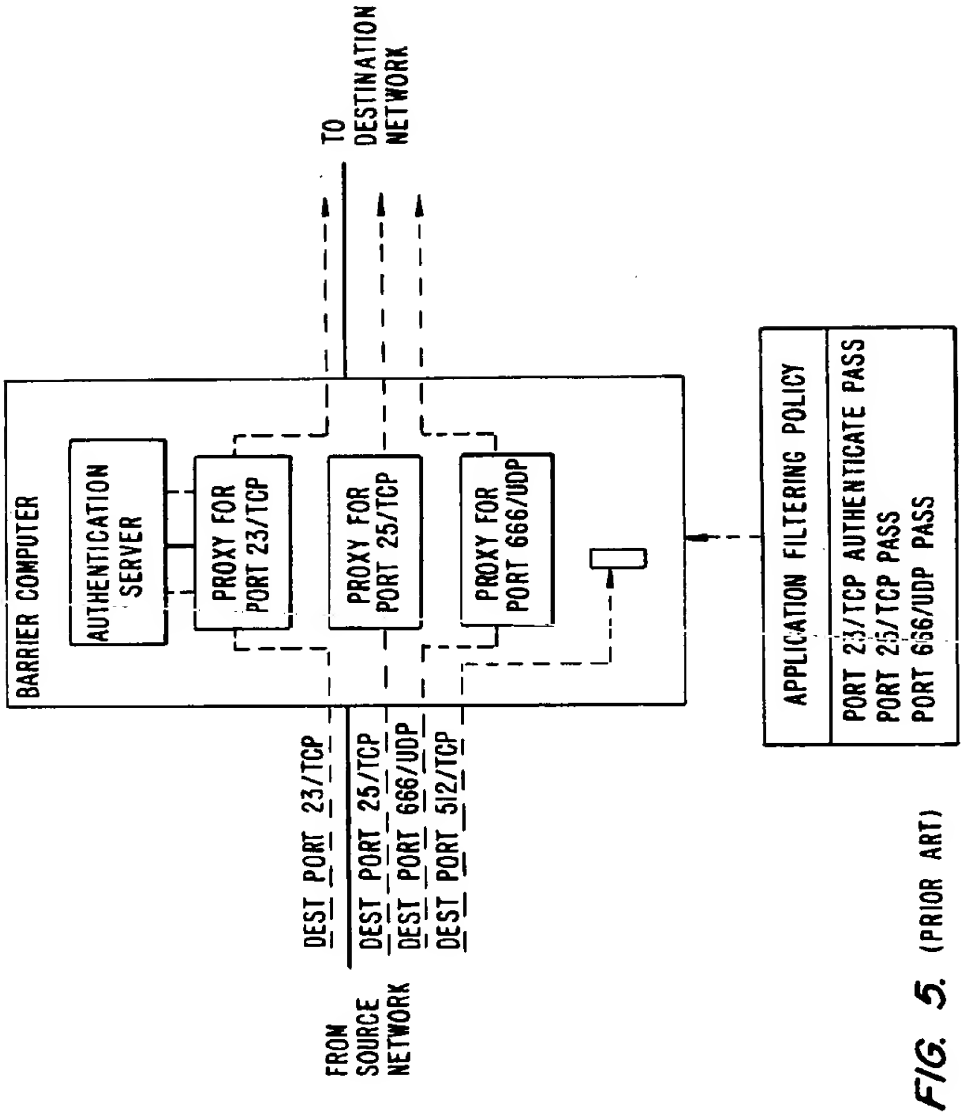



FIG. 5. (PRIOR ART)

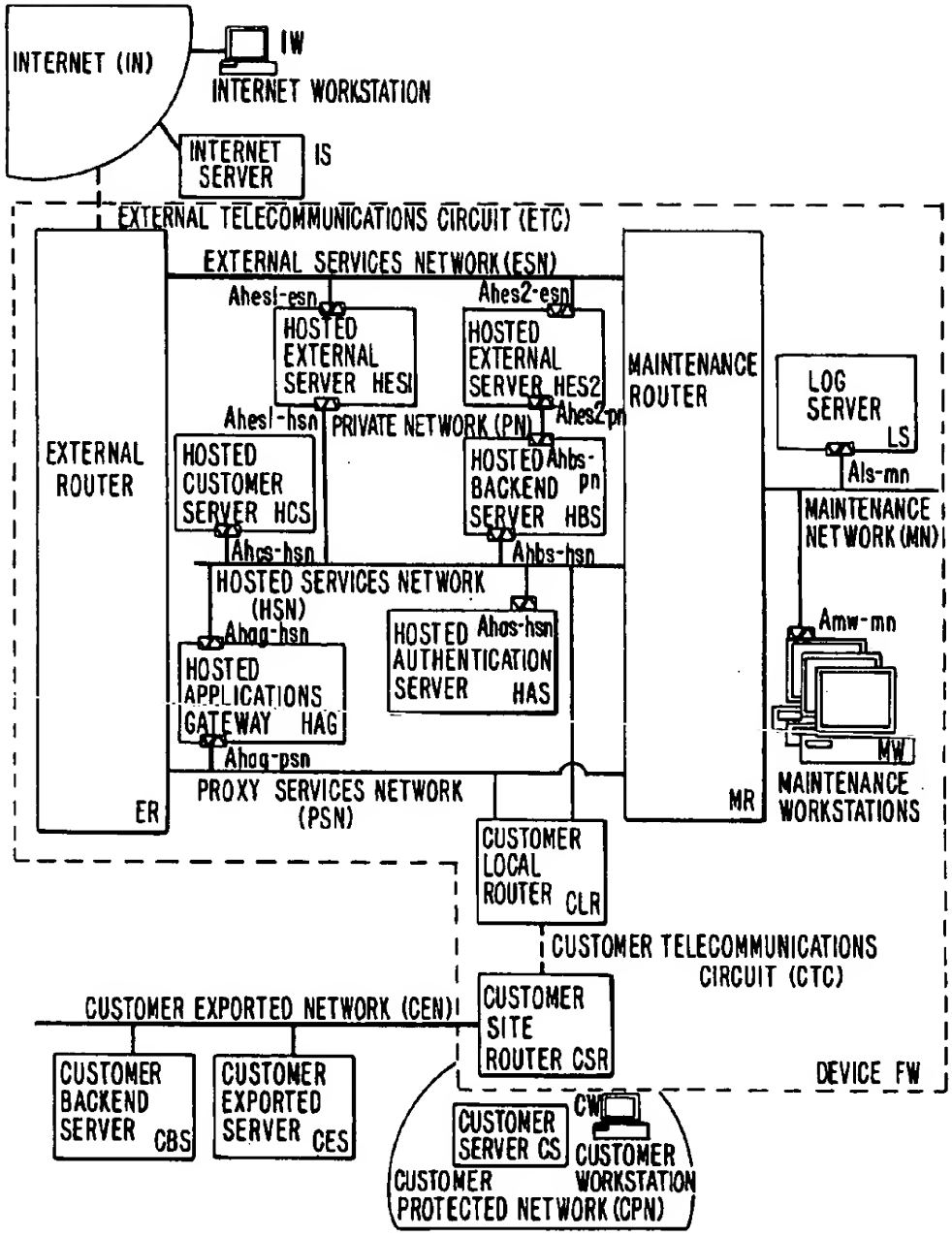

INTERNET (IN)
IW
INTERNET WORKSTATION
INTERNET SERVER
IS
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
EXTERNAL SERVICES NETWORK (ESN)
Ahes1-esn
Ahes2-esn
HOSTED EXTERNAL SERVER HES1
HOSTED EXTERNAL SERVER HES2
MAINTENANCE ROUTER
LOG SERVER LS
Ahes1-hsn
PRIVATE NETWORK (PN)
Ahes2-pn
EXTERNAL ROUTER
HOSTED CUSTOMER SERVER HCS
HOSTED BACKEND SERVER HBS
Ahbs-pn
Als-mn
MAINTENANCE NETWORK (MN)
Ahcs-hsn
Ahbs-hsn
HOSTED SERVICES NETWORK (HSN)
Amw-mn
Ahag-hsn
HOSTED AUTHENTICATION SERVER HAS
Ahas-hsn
HOSTED APPLICATIONS GATEWAY HAG
MW
Ahag-psn
PROXY SERVICES NETWORK (PSN)
ER
MR
MAINTENANCE WORKSTATIONS
CUSTOMER LOCAL ROUTER CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
CUSTOMER EXPORTED NETWORK (CEN)
CUSTOMER SITE ROUTER CSR
DEVICE FW
CUSTOMER BACKEND SERVER CBS
CUSTOMER EXPORTED SERVER CES
CUSTOMER SERVER CS
CW
CUSTOMER WORKSTATION
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 6.

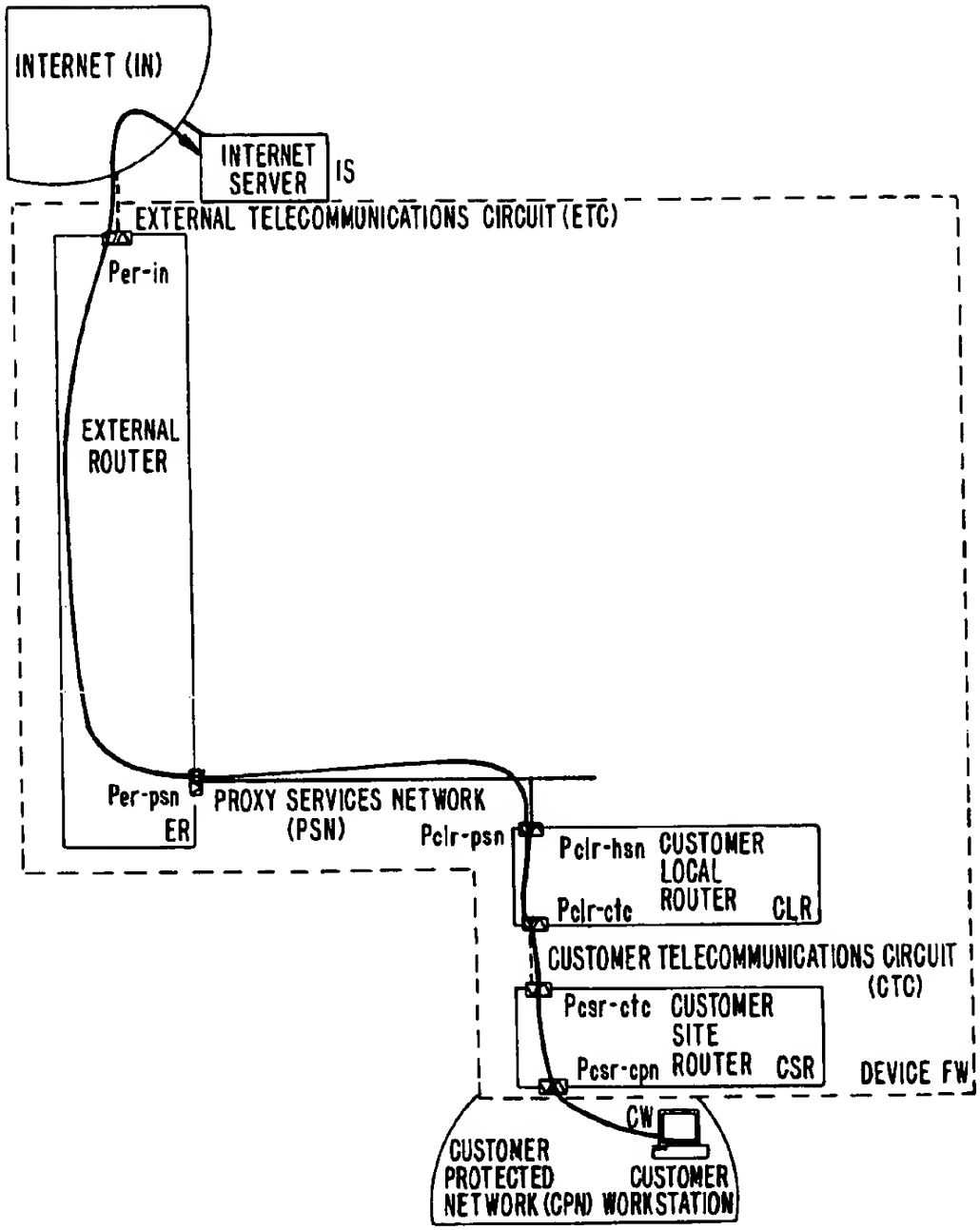

INTERNET (IN)
INTERNET SERVER
IS
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
Per-in
EXTERNAL ROUTER
Per-psn
ER
PROXY SERVICES NETWORK (PSN)
Pclr-psn
Pclr-hsn
CUSTOMER LOCAL ROUTER
CLR
Pclr-ctc
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
Pcsr-ctc
CUSTOMER SITE ROUTER
CSR
Pcsr-cpn
DEVICE FW
CW
CUSTOMER PROTECTED NETWORK (CPN)
CUSTOMER WORKSTATION

FIG. 7.

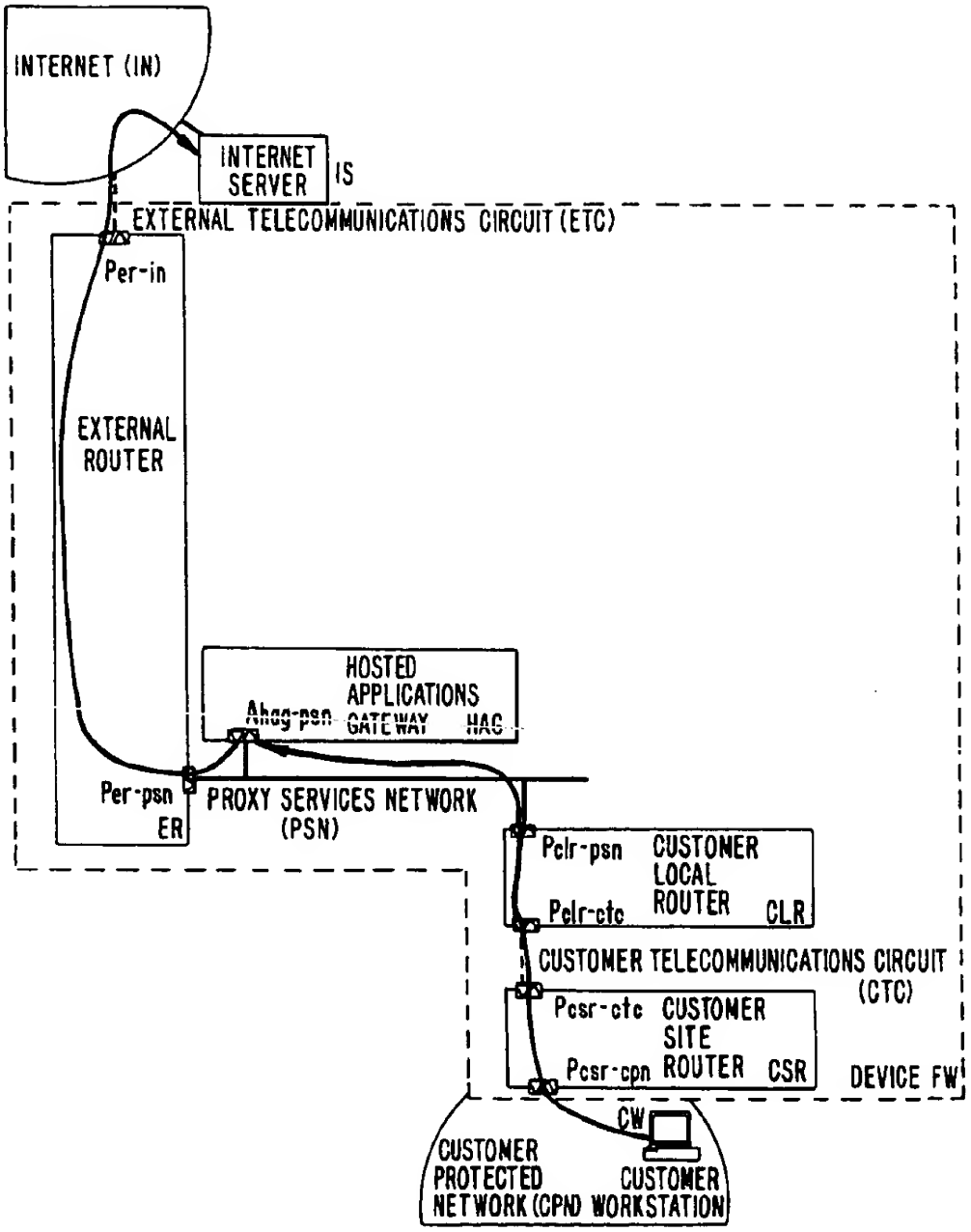

INTERNET (IN)
INTERNET SERVER
IS
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
Per-in
EXTERNAL ROUTER
HOSTED APPLICATIONS
Ahag-psn
GATEWAY
HAG
Per-psn
ER
PROXY SERVICES NETWORK (PSN)
Pclr-psn
CUSTOMER LOCAL ROUTER
Pclr-ctc
CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
Pcsr-ctc
CUSTOMER SITE ROUTER
Pcsr-cpn
CSR
DEVICE FW
CW
CUSTOMER PROTECTED NETWORK (CPN)
CUSTOMER WORKSTATION

FIG. 8.

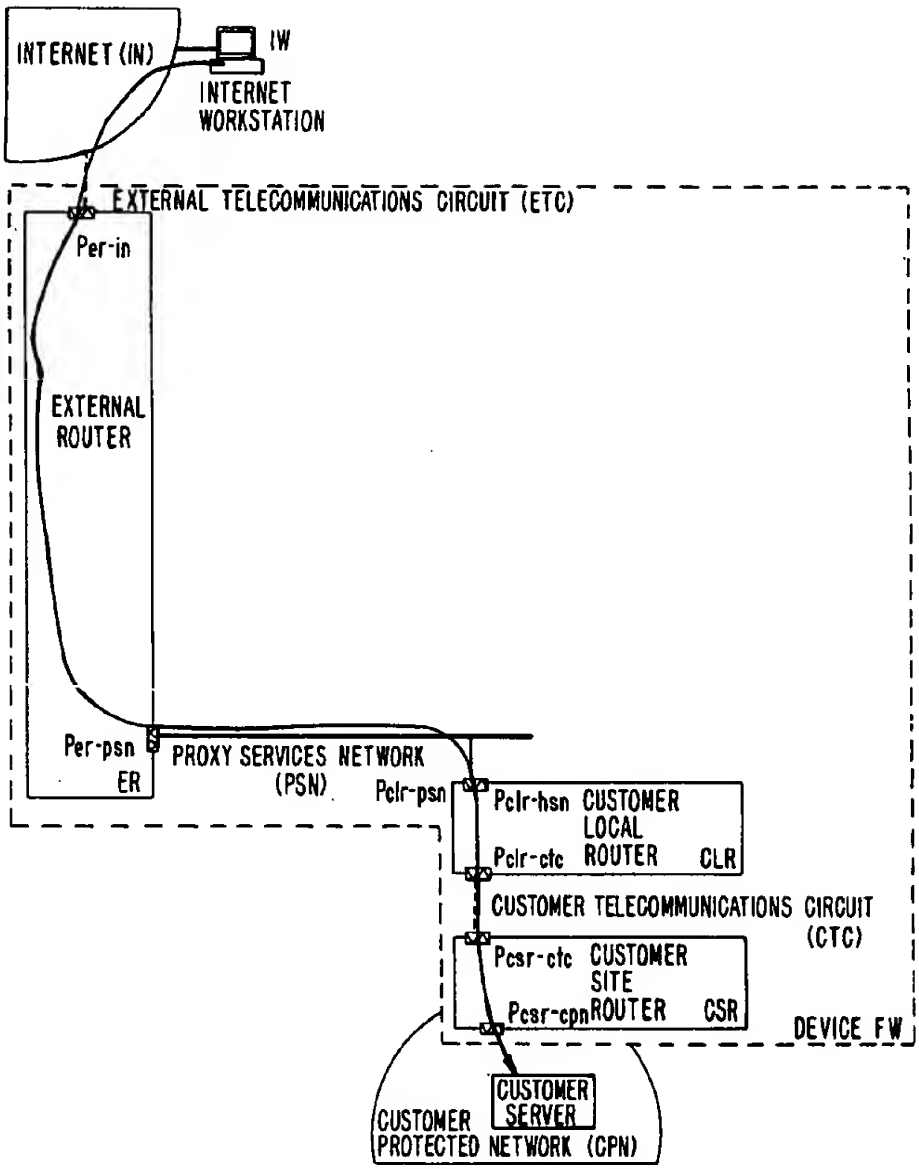

INTERNET (IN)
IW
INTERNET WORKSTATION
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
Per-in
EXTERNAL ROUTER
Per-psn
ER
PROXY SERVICES NETWORK (PSN)
Pclr-psn
Pclr-hsn CUSTOMER LOCAL
Pclr-ctc ROUTER CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
Pcsr-ctc CUSTOMER SITE
Pcsr-cpn ROUTER CSR
DEVICE FW
CUSTOMER SERVER
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 9.

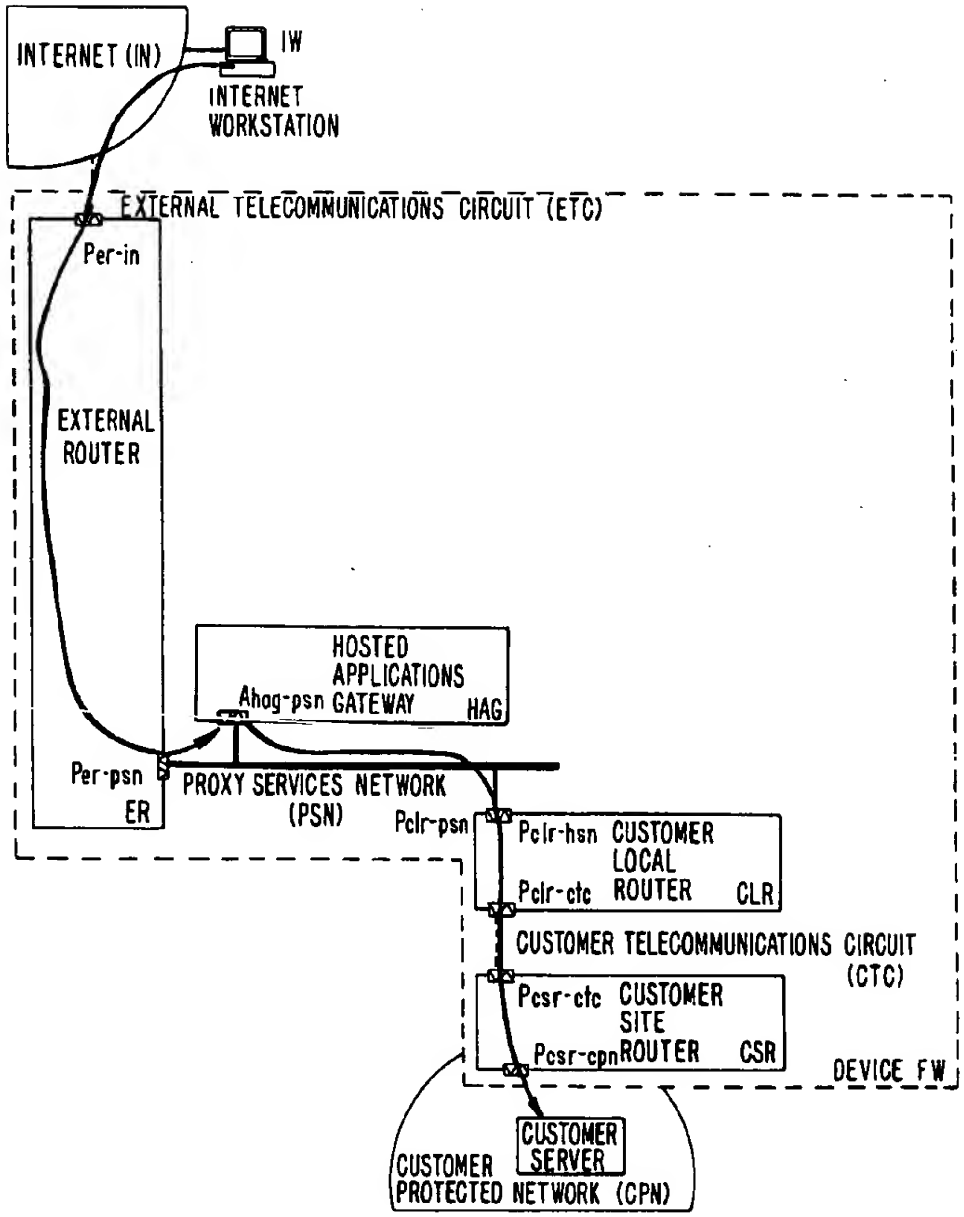

INTERNET (IN)
IW
INTERNET WORKSTATION
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
Per-in
EXTERNAL ROUTER
HOSTED APPLICATIONS
Ahag-psn GATEWAY HAG
Per-psn ER
PROXY SERVICES NETWORK (PSN)
Pclr-psn
Pclr-hsn CUSTOMER LOCAL
Pclr-ctc ROUTER CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
Pcsr-ctc CUSTOMER SITE
Pcsr-cpn ROUTER CSR
DEVICE FW
CUSTOMER SERVER
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 10.

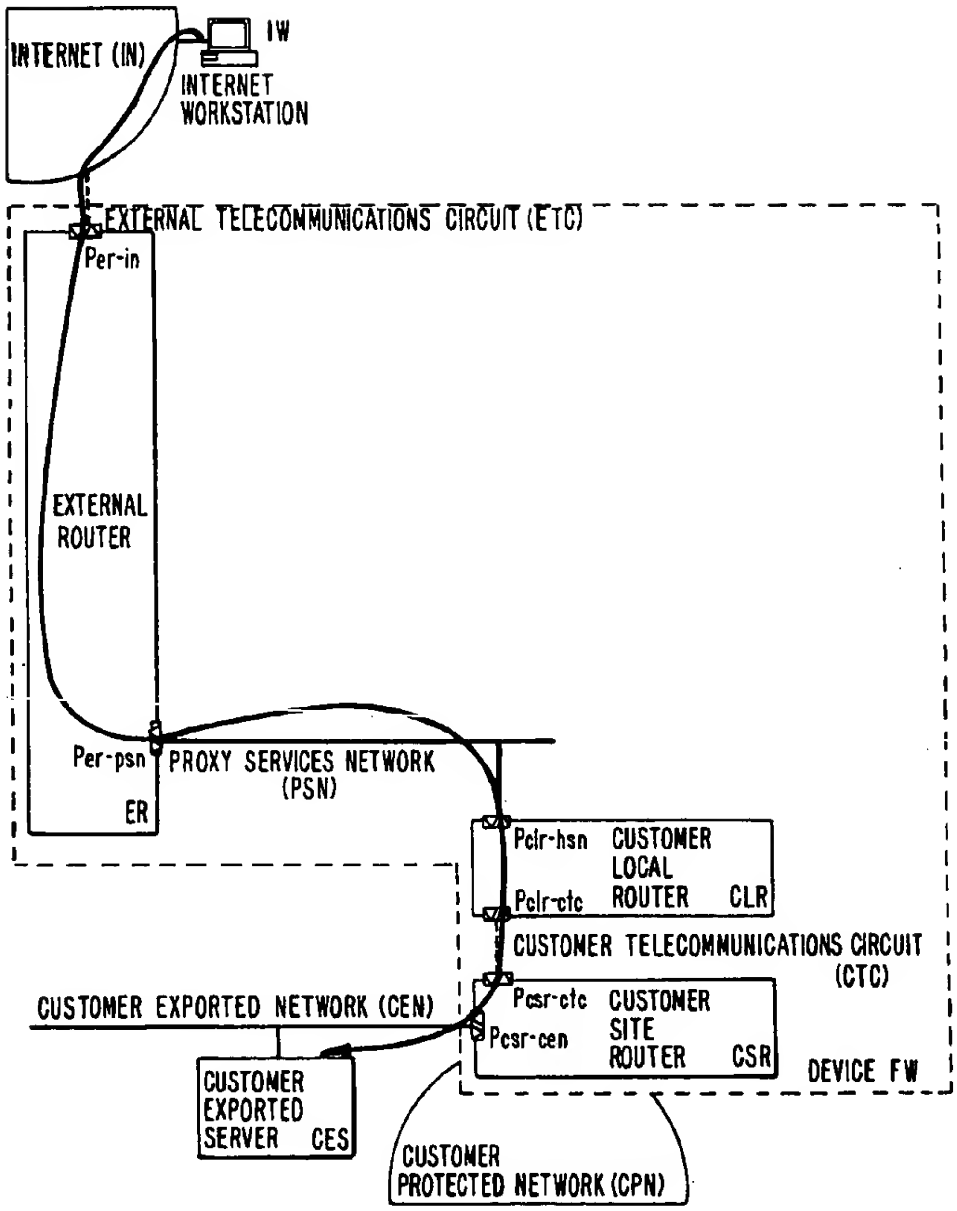

INTERNET (IN)
IW
INTERNET WORKSTATION
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
Per-in
EXTERNAL ROUTER
Per-psn
ER
PROXY SERVICES NETWORK (PSN)
Pclr-hsn
CUSTOMER LOCAL ROUTER
CLR
Pclr-ctc
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
CUSTOMER EXPORTED NETWORK (CEN)
Pcsr-ctc
Pcsr-cen
CUSTOMER SITE ROUTER
CSR
DEVICE FW
CUSTOMER EXPORTED SERVER
CES
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 11.

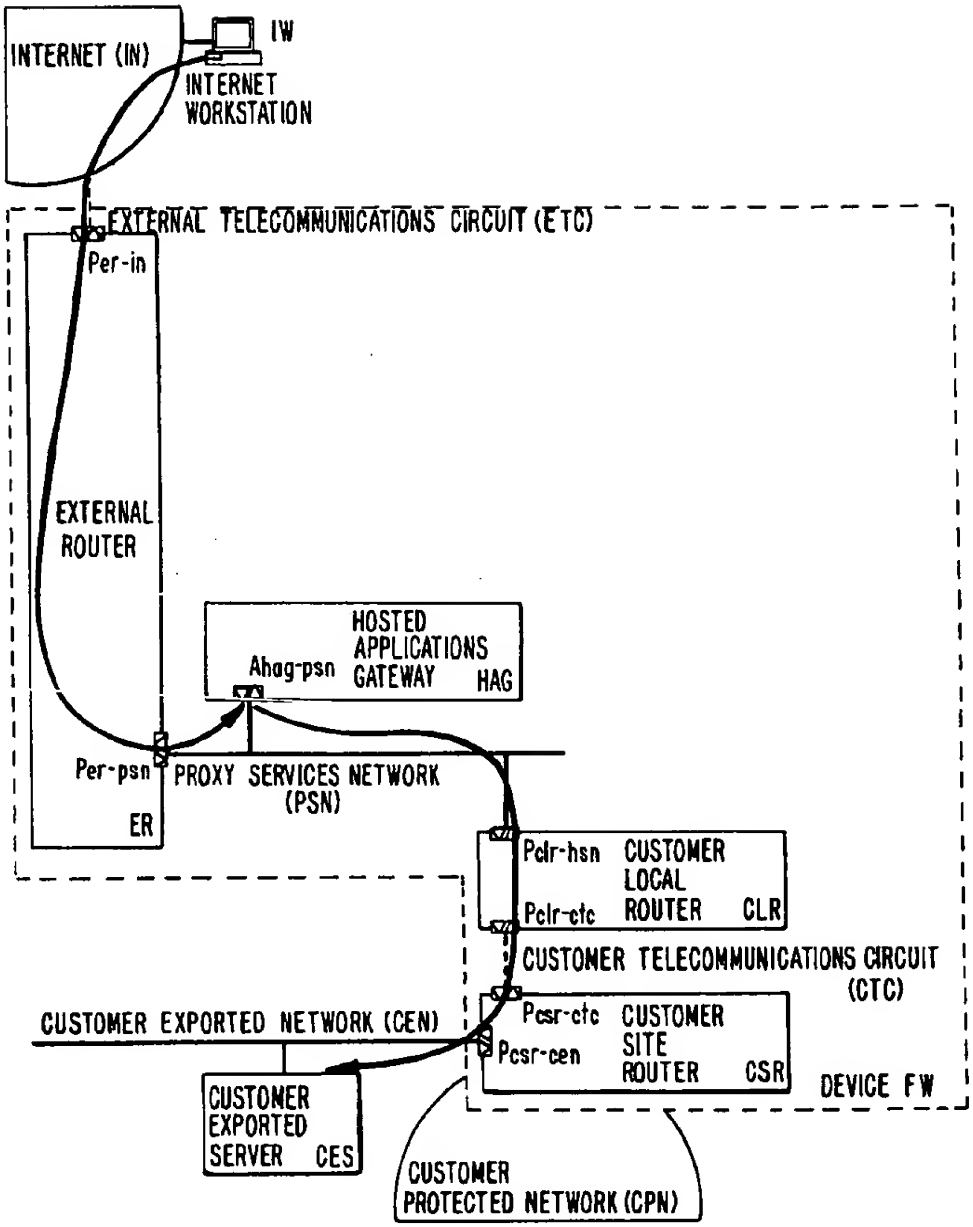

INTERNET (IN)
IW
INTERNET WORKSTATION
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
Per-in
EXTERNAL ROUTER
HOSTED APPLICATIONS GATEWAY HAG
Ahag-psn
Per-psn
ER
PROXY SERVICES NETWORK (PSN)
Pclr-hsn
Pclr-ctc
CUSTOMER LOCAL ROUTER CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
CUSTOMER EXPORTED NETWORK (CEN)
Pcsr-ctc
Pcsr-cen
CUSTOMER SITE ROUTER CSR
DEVICE FW
CUSTOMER EXPORTED SERVER CES
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 12.

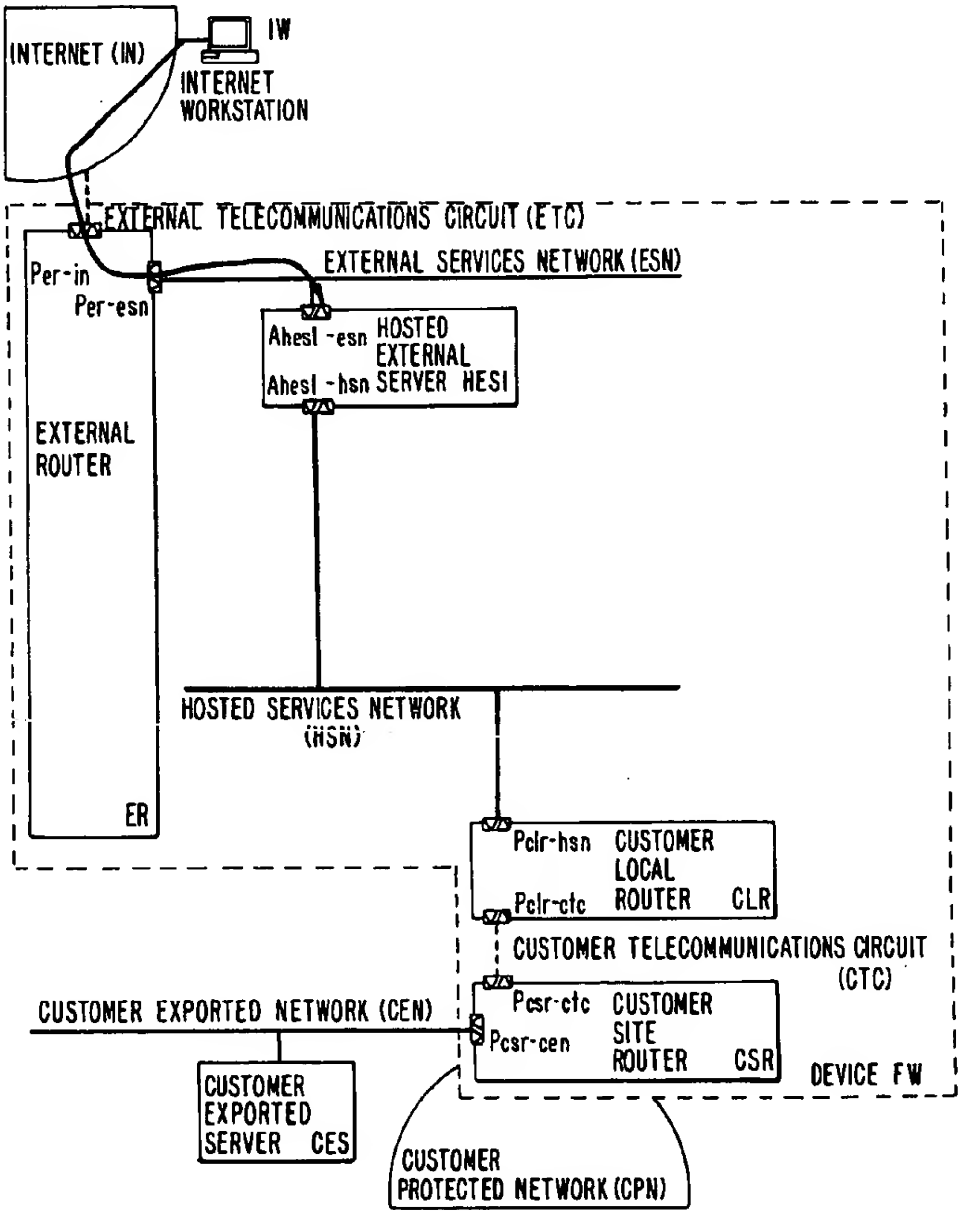

INTERNET (IN)
IW
INTERNET WORKSTATION
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
EXTERNAL SERVICES NETWORK (ESN)
Per-in
Per-esn
Ahesl-esn
Ahesl-hsn
HOSTED EXTERNAL SERVER HESI
EXTERNAL ROUTER
ER
HOSTED SERVICES NETWORK (HSN)
Pclr-hsn
Pclr-ctc
CUSTOMER LOCAL ROUTER CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
Pcsr-ctc
Pcsr-cen
CUSTOMER SITE ROUTER CSR
DEVICE FW
CUSTOMER EXPORTED NETWORK (CEN)
CUSTOMER EXPORTED SERVER CES
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 13.

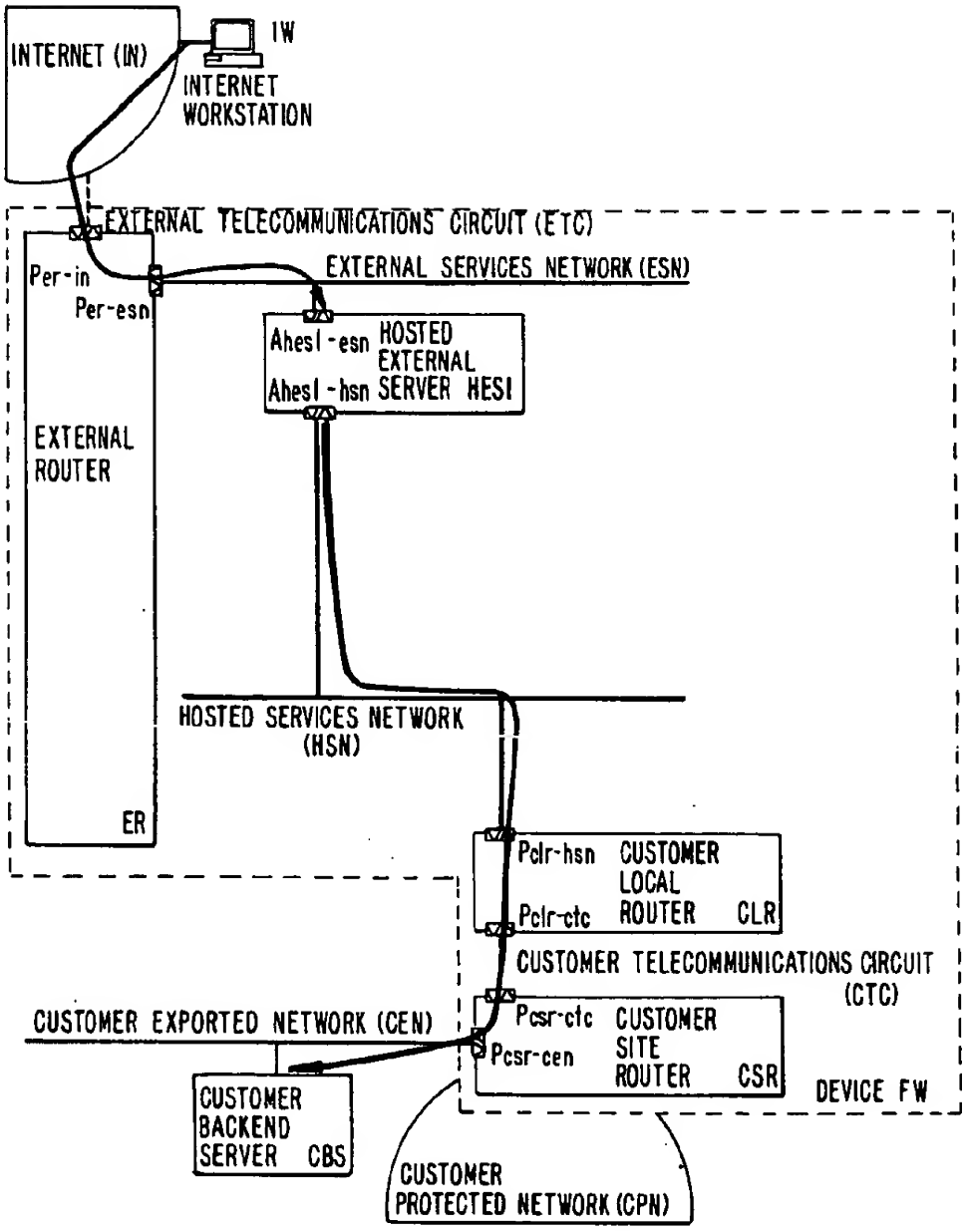

INTERNET (IN)
IW
INTERNET WORKSTATION
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
EXTERNAL SERVICES NETWORK (ESN)
Per-in
Per-esn
Ahesl-esn
Ahesl-hsn
HOSTED EXTERNAL SERVER HESI
EXTERNAL ROUTER
ER
HOSTED SERVICES NETWORK (HSN)
Pclr-hsn
Pclr-ctc
CUSTOMER LOCAL ROUTER CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
CUSTOMER EXPORTED NETWORK (CEN)
Pcsr-ctc
Pcsr-cen
CUSTOMER SITE ROUTER CSR
DEVICE FW
CUSTOMER BACKEND SERVER CBS
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 14.

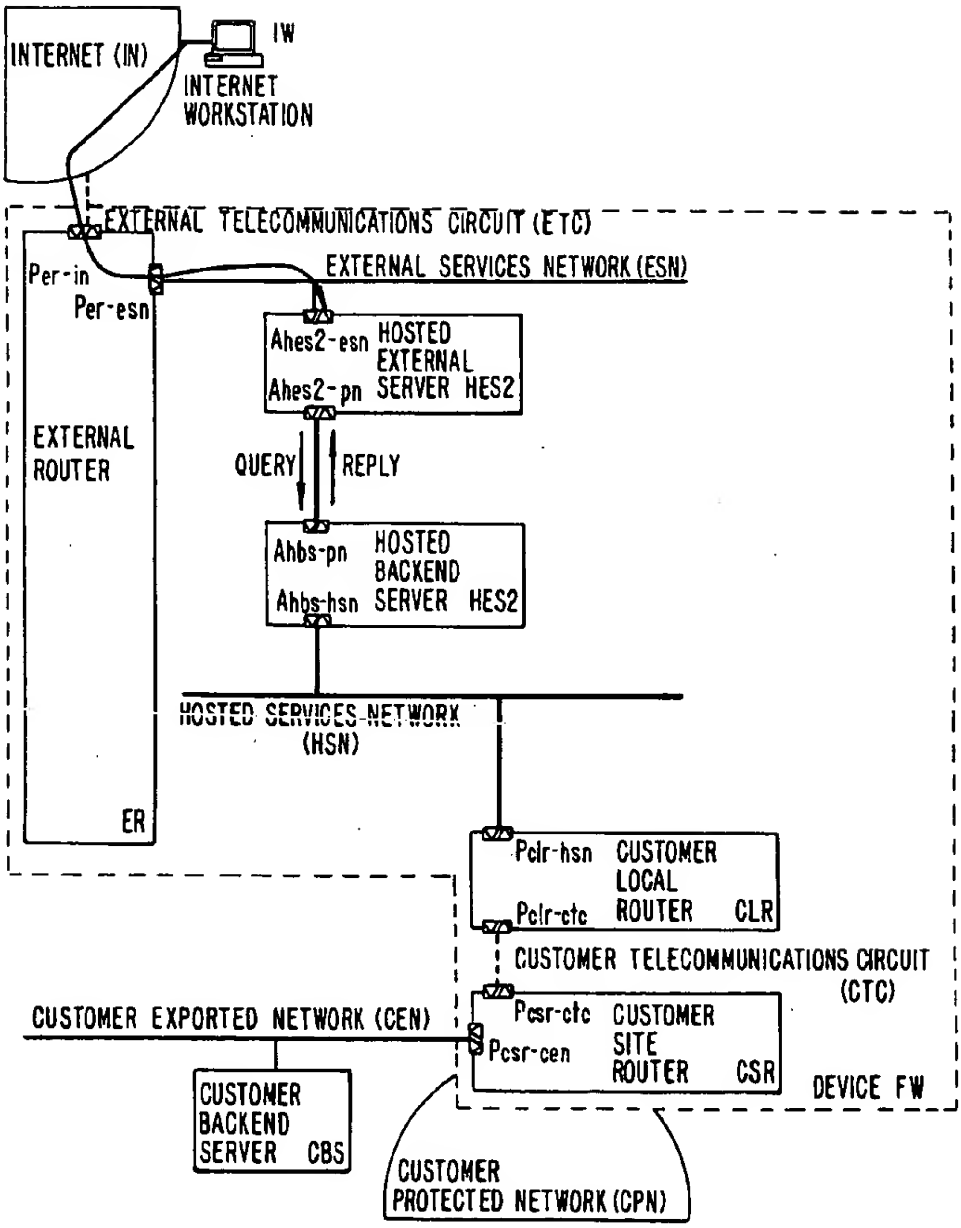

INTERNET (IN)
IW
INTERNET WORKSTATION
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
EXTERNAL SERVICES NETWORK (ESN)
Per-in
Per-esn
EXTERNAL ROUTER
ER
Ahes2-esn
Ahes2-pn
HOSTED EXTERNAL SERVER HES2
QUERY
REPLY
Ahbs-pn
Ahbs-hsn
HOSTED BACKEND SERVER HES2
HOSTED SERVICES NETWORK (HSN)
Pclr-hsn
Pclr-ctc
CUSTOMER LOCAL ROUTER CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
Pcsr-ctc
Pcsr-cen
CUSTOMER SITE ROUTER CSR
DEVICE FW
CUSTOMER EXPORTED NETWORK (CEN)
CUSTOMER BACKEND SERVER CBS
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 15.

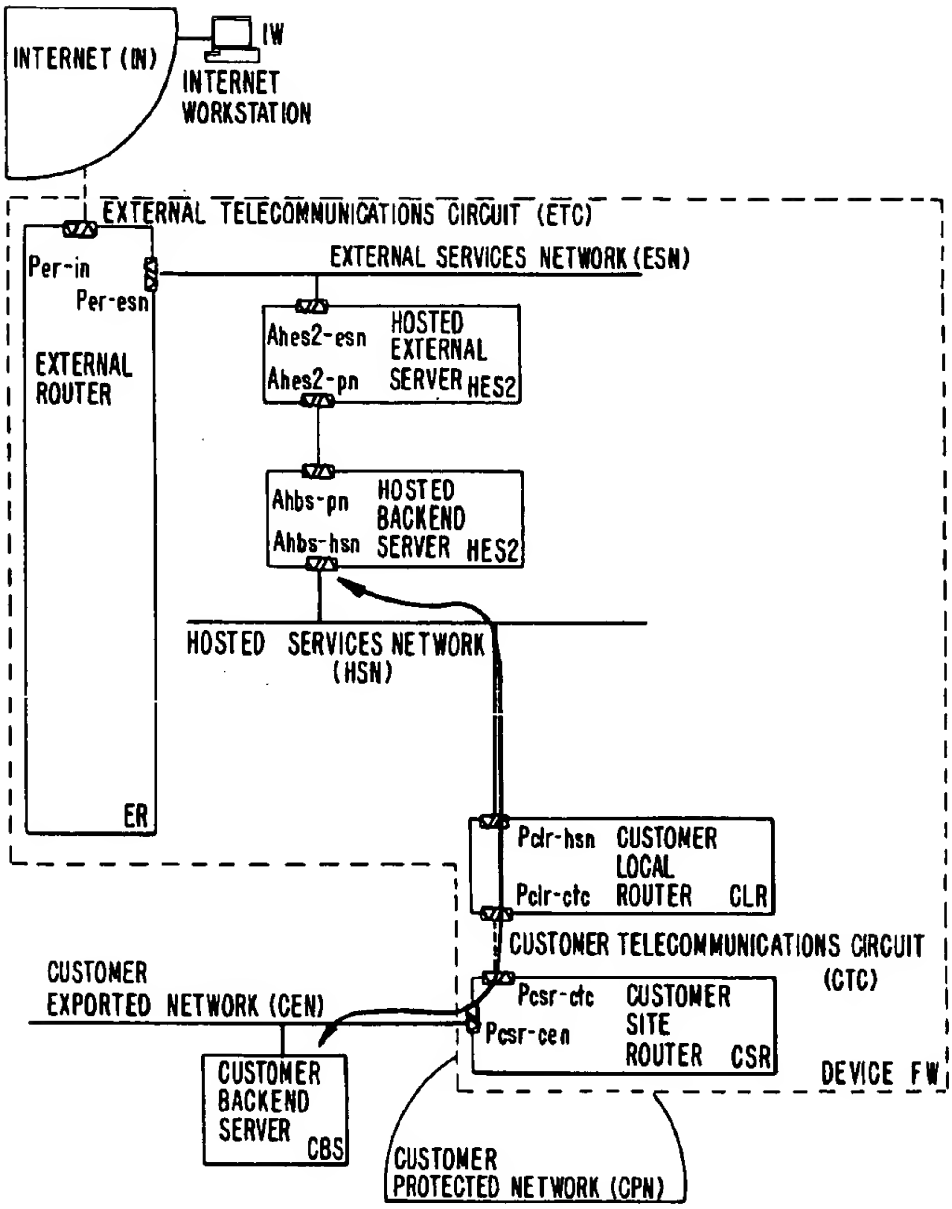

INTERNET (IN)
IW
INTERNET WORKSTATION
EXTERNAL TELECOMMUNICATIONS CIRCUIT (ETC)
EXTERNAL SERVICES NETWORK (ESN)
Per-in
Per-esn
EXTERNAL ROUTER
ER
Ahes2-esn
Ahes2-pn
HOSTED EXTERNAL SERVER HES2
Ahbs-pn
Ahbs-hsn
HOSTED BACKEND SERVER HES2
HOSTED SERVICES NETWORK (HSN)
Pclr-hsn
Pclr-ctc
CUSTOMER LOCAL ROUTER CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
CUSTOMER EXPORTED NETWORK (CEN)
Pcsr-ctc
Pcsr-cen
CUSTOMER SITE ROUTER CSR
DEVICE FW
CUSTOMER BACKEND SERVER CBS
CUSTOMER PROTECTED NETWORK (CPN)

FIG. 16.

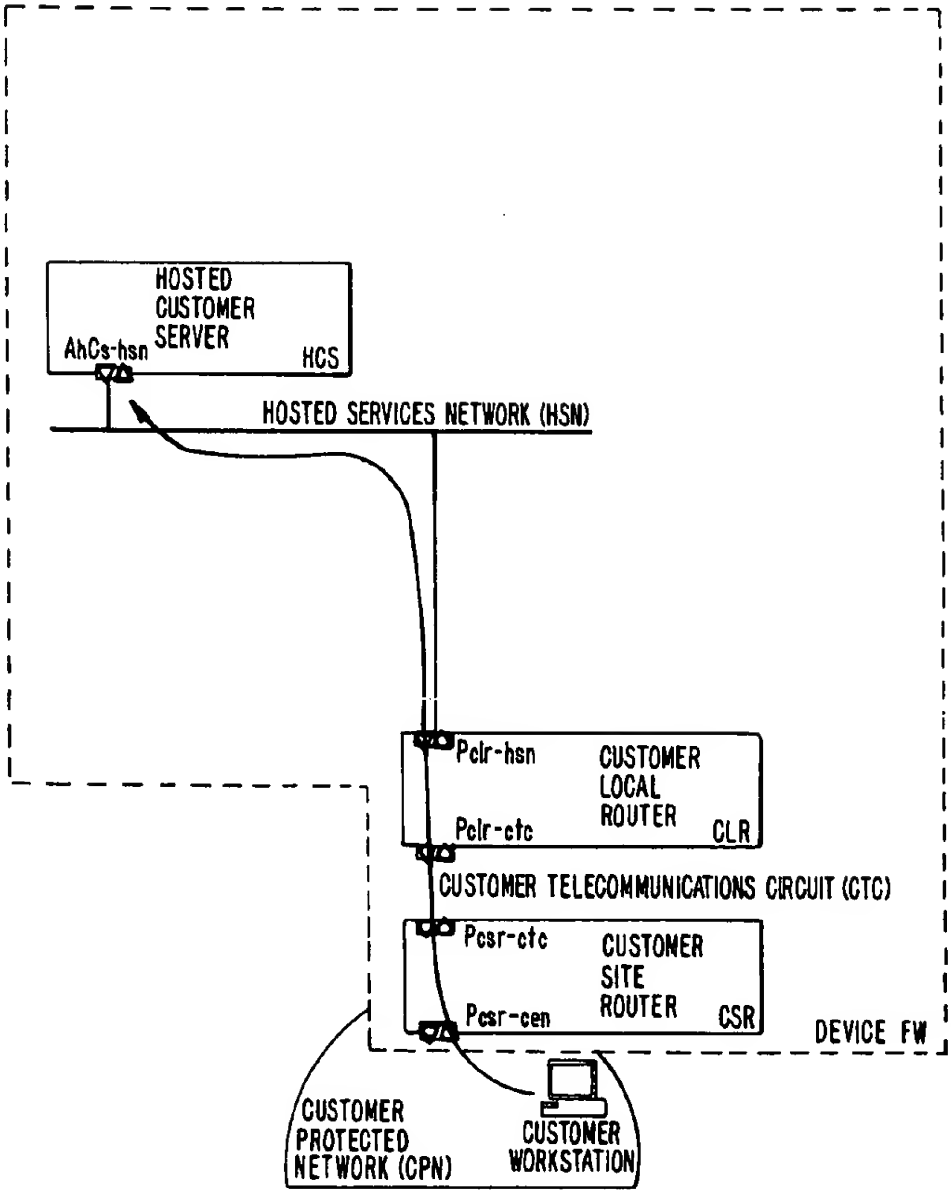

HOSTED
CUSTOMER
SERVER
AhCs-hsn
HCS
HOSTED SERVICES NETWORK (HSN)
Pclr-hsn
CUSTOMER
LOCAL
ROUTER
Pclr-ctc
CLR
CUSTOMER TELECOMMUNICATIONS CIRCUIT (CTC)
Pcsr-ctc
CUSTOMER
SITE
ROUTER
Pcsr-cen
CSR
DEVICE FW
CUSTOMER
PROTECTED
NETWORK (CPN)
CUSTOMER
WORKSTATION

FIG. 17.

# DYNAMIC POLICY-BASED APPARATUS FOR WIDE-RANGE CONFIGURABLE NETWORK SERVICE AUTHENTICATION AND ACCESS CONTROL USING A FIXED-PATH HARDWARE CONFIGURATION

## COPYRIGHT NOTICE



## BACKGROUND OF THE INVENTION

This invention relates to internetwork communications and data exchanges, and in particular to the security of information exchange between computer networks to inhibit and detect attempts at vandalism, espionage, sabotage or inadvertent destruction of data.

Computer networks connect multiple computer systems together, allowing them to share information. Initially, the computers were in one, secure location. As the utility of networks grew, it became more and more desirable to connect networks at different locations to allow information to flow between the computer systems at all sites. As the number of computer systems grew beyond the point where each user of the network was well known to all other users, the need for a mechanism to describe and enforce a policy for access, known as a "security policy", became apparent.

Two major techniques developed for security policy enforcement. The first is packet filtering, in which a security policy specifies what types of connections are allowed and permits or denies passage of TCP/IP packets of specific types through a router. The second technique is application filtering, which operates at a higher level, examining the specific transactions that pass through a TCP/IP connection, and allowing them or denying them based on the specific action being attempted, or the identity of the requester.

When combined, these two techniques comprise a firewall, whose purpose is to implement and enforce the security policy of an organization regarding connections between two or more networks. Historically, there have been two major types of firewalls: custom and commercial. A custom firewall is a device or collection of devices designed, purchased, assembled, configured and operated by an organization for the purposes of guarding a network interconnection. A commercial firewall collects many of the components of a custom firewall into a single device, and is sold (and sometimes configured) by a company to make the installation of a firewall easier and more cost effective.

Custom firewalls have many potential drawbacks. For one, because they are designed and constructed by a single organization that may not have extensive experience in the problems of firewall design, they may not account for many known problems. Because they are designed and built for a specific purpose, they are typically very difficult to adapt to new policies. This often requires a significant redesign effort, and additional hardware. Because they are built in a unique manner, each custom firewall requires special software, special training, and special expertise in modifications that does not translate into other firewall installations.

Commercial firewalls are designed to consolidate as many services as possible into a single box. That box is then used as the focus of a customer-specific firewall. Because some services, such as packet filtering, are often done better in routers, commercial firewalls are rarely used by themselves. Additional devices, and a design for their use, is often required, which returns the customer to many of the problems inherent in a custom firewall.

In addition, commercial firewalls are configured by the user, who may be unaware of many of the issues and problems of security policy design. It is estimated that more than 30% of all firewall penetrations happen through a commercial or custom firewall. This is typically because of poorly thought out configuration.

Because much of the functionality of a commercial firewall is concentrated in a single box, these devices also invite other problems. If the device fails, all communication between networks is cut off. There is no ability to gracefully degrade service. If the security of one service of the box is compromised, this can open a path for an attacker to compromise other services and widen their access. Also, the design of the single-box firewall very strongly affects the types of policies available to the customer. If a box is designed primarily as an applications gateway device, it is very difficult to configure it in a firewall that will permit some services to be performed via packet-filtering only.

One problem that is shared by both types of firewalls is that of scalability. Because each type of firewall has strong hardware/software/configuration customizations for each specific customer, managing the firewalls of more than one customer is very difficult. Making significant policy changes in multiple customer firewalls is also extremely difficult.

Because of these scalability problems, it has been quite difficult for a company to offer managed firewall services to many customers, since the scaling problems escalate with each new customer.

## SUMMARY OF THE INVENTION

An improved security handler is provided by virtue of the present invention. In one embodiment of a security handler according to the present invention, a security handler includes means for obtaining customer security policies, a plurality of packet processing components with communications paths therebetween and configurable policy enforcement means, for enforcing a packet policy over the communications paths.

One advantage of the present invention is that a single configuration of physical components can be configured to provide a wide range of security policy choices while remaining capable of solving the foregoing problems of the prior art.

A further understanding of the nature and advantages of the inventions herein may be realized by reference to the remaining portions of the specification and the attached drawings.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a security handler implementation abbreviated herein as "Device FW".

FIG. 2 is a schematic of a TCP/IP packet.

FIGS. 3, 3*a* show a representation of the process of packet filtering.

FIG. 4 shows the points within Device FW with representative points at which a packet filtering policy can be established.

FIG. 5 shows an example of an applications level filter.

FIG. 6 shows the points within Device FW with representative points at which an application filtering policy can be established.

FIG. 7 is a simplified block diagram of Device FW showing a path for traffic between a workstation located on a Customer Protected Network (CPN) and a server on the Internet via a directly-routed, packet-filtered path.

FIG. 8 is a simplified block diagram of Device FW showing a path for traffic between a workstation on the CPN and an Internet server via an application-filtered connection.

FIG. 9 is a simplified block diagram of Device FW showing a path for traffic between a workstation on the Internet and a server on the CPN via a directly-routed, packet-filtered connection.

FIG. 10 is a simplified block diagram of Device FW showing a path for traffic between a workstation on the Internet and a server on the CPN via an application-filtered connection.

FIG. 11 is a simplified block diagram of Device FW showing a path for traffic between a workstation on the Internet and a server on a Customer Exported Network (CEN) via a directly routed, packet-filtered connection.

FIG. 12 is a simplified block diagram of Device FW showing a path for traffic between a workstation on the Internet and a server on the CEN via an application-filtered connection.

FIG. 13 is a simplified block diagram of Device FW showing a path for traffic between an Internet workstation and a server hosted within Device FW.

FIG. 14 is a simplified block diagram of Device FW showing another path for traffic between an Internet workstation and a server hosted within Device FW.

FIG. 15 is a simplified block diagram of Device FW showing a yet another path for traffic between an Internet workstation and a server hosted within Device FW.

FIG. 16 is a simplified block diagram of Device FW showing a path a replication connection would use to copy data from a Customer Backend Server (CBS) to a Hosted Backend Server (HBS).

FIG. 17 is a simplified block diagram of Device FW showing a path for traffic from a workstation on the CPN for directly-routed, packet-filtered access to a Hosted Customer Server.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

### FIG. 1—System Overview

FIG. 1 shows the components of Device FW and the way in which each component is interconnected. Also shown, is an example external network (the Internet) and an example internal network (the Customer Network) to clarify the relationship between FW and an external and internal network.

In the preferred embodiment, Device FW is a collection of components, physically connected by networks and telecommunications circuits in an arrangement which permits a safe, but flexible range of usage strategies. Networks allow one component to communicate with another component in close physical proximity and the network can be form of multi-point connectivity, such as Ethernet, Fast Ethernet, FDDI, ATM or other networks that provide similar functionality. In the preferred embodiment described herein, network traffic is based on the TCP/IP network protocol.

FW communicates with components that are not in close physical proximity using telecommunications circuits. A telecommunications circuit can be any point-to-point link capable of carrying TCP/IP traffic, such as a T-1 line, a 56K line, a modem-based telephone connection, a microwave relay, a fiber-optic circuit.

TCP/IP packets are moved between networks or between telecommunications circuits and networks, or between different telecommunications circuits, by routers. Routers are also used for applying TCP/IP packet-filtering rules, which permit or deny passage to packets based on characteristics such as origination address, destination address, origination service port, destination service port, protocol, size, contents, time of day, encryption, etc. Preferably, each router supports a packet filtering model comparable to that of Cisco Systems IOS version 11.0 or later. Routers are available from various manufacturers such as. Cisco, Bay Network, or 3Com.

Servers are used to provide information to a network or collect information from a network. A server provides a service which is made available to workstations across a network. Servers can come from a variety of manufacturers, such as Sun Microsystems, Hewlett Packard, Silicon Graphics Digital Equipment Corporation. Servers run a variety of operating systems such as Solaris, SunOS, HP/UX, BSDI, Linux, FreeBSD. Preferably, whatever operating system is used, it should have the ability to eliminate extraneous software and services, control access to services and log access to services.

Servers can provide many types of service, such as World-Wide Web (WWW), File Transfer (FTP), RealAudio, RealVideo, authentication, database, logging. Gateways are used to apply a security policy to an application's network connection. Workstations are used to connect human beings to servers, and are also used to allow maintainers of Device FW to inspect and modify the security policy of Device FW.

Referring now to FIG. 1, the description of several individual components is now provided. An External Router (ER) controls traffic to and from the External network and enforces the outermost layer of the Security Policy. A Maintenance Router (MR) connects the portions of Device FW that implement the security policy with the Maintenance Network that host servers use to record the behavior of FW and workstations used to inspect or modify the behavior of FW.

The Customer Local Router (CLR) connects Device FW to the Customer Site Router (CSR). The CLR also limits the connections to and from a customer. The CSR connects a Customer Exported Network (CEN) and the Customer Protected Network (CPN). The CSR defines what types of traffic can pass between the CPN and Device FW, between the CEN and Device FW and between the CEN and the CPN.

A Log Server (LS) is used by FW to record the state and behavior of components. From this recorded information, statistics can be gathered about attacks, performance, connections established, and the success or failure of particular portions of the Security Policy. A Maintenance Workstation (MW) is used to inspect or change the behavior of Device FW.

A typical FW contains most or all of the following types of components:

Networks
Telecommunications circuits
Routers
Servers
Gateways
Workstations

The embodiment of FW shown in FIG. 1 is shown including the following components:

ER—External Router

MR—Maintenance Router

CLR—Customer Local Router

CSR—Customer Site Router

LS—Log Server

MW—Maintenance Workstation

In addition, FIG. 1 shows the following representative components, of which there may be one or more actual components in an implementation (only one of each of these are shown for clarity and to illustrate the principles involved):

HCS—Hosted Customer Server

HES1—Hosted External Server, Type 1

HES2—Hosted External Server, Type 2

HBS—Hosted Backend Server

HAG—Hosted Applications Gateway

HAS—Hosted Authentication Server

As shown in FIG. 1, FW also includes the following networks and telecommunications circuits:

ESN—External Services Network

HSN—Hosted Services Network

PSN—Proxy Services Network

ETC—External Telecommunications Circuit

CTC—Customer Telecommunications Circuit

The External Network (EN) is shown including:

ETC—External Telecommunications Circuit

IN—Internet Network

IS—Internet Server

IW—Internet Workstation

Also shown for purposes of clarification is a Customer Networks, which includes:

CS—Customer Server

CW—Customer Workstation

CPN—Customer Protected Network

CEN—Customer Exported Network

CES—Customer Exported Server

CBS—Customer Backend Server

There may be any number of CBS and CES systems connected to the CEN. One of each is shown for purposes of clarification. ER is connected to ETC, ESN and PSN. MR is connected to ESN, HSN, PSN and MN. CLR is connected to CTC, PSN and HSN. CSR is connected to CTC, CPN and CEN. ESN is connected to ER, MR, HES1, and HES2. HSN is connected to MR, CLR, HCS, HES1, HBS, HAG and HAS. PSN is connected to ER, MR and HAG. MN is connected to MR, LS and MW. CEN is connected to CSR, CBS, and CES.

FIG. 2 shows a representation of a TCP/IP packet. A TCP/IP packet is a sequence of bits that is transmitted across a network or telecommunications circuit. Computers connected to the network can transmit or receive TCP/IP packets in order to implement a variety of services (such as remote terminal sessions, file transfer, and electronic mail). A service may require one or more packets to be transmitted and/or received.

A TCP/IP packet includes an "IP" layer and a "Payload" layer. The IP layer is used as an "envelope" to hold the Payload layer. The IP layer contains the address of the source machine (the "transmitter" of the packet), the address of the destination machine (the "receiver" of the packet) and other information pertaining to the delivery of the packet to its intended recipient. All of this information is kept in fields within the packet. The payload portion of the IP packet contains information pertaining to the specific application using the packet. This information is organized as either a TCP packet, a UDP packet or an ICMP packet.

A TCP packet is used to carry part of a connection that is "stream-like". This would be information that is in a sequence, such as the contents of a file or a virtual terminal session. TCP packets are in sequence, and are "reliable", which means that they are guaranteed to arrive in order if they arrive at all. If they do not arrive, their absence will be noticed. A UDP packet is intended for data that is nonsequential. UDP packets are often called "datagrams" to emphasize their similarity to telegrams, in which the entire context of the message is contained within the single packet or message. A UDP packet might be used for an occasional status update or to log a particular event. UDP packets are nonsequential, and arrive in no predefined order. They are "unreliable" in that a packet may not arrive and if it does not its absence will not be noticed. An ICMP packet is used for network signalling. In particular, ICMP messages are used to indicate a problem of some kind with the transmission of packets from their source to their destination. Additionally, ICMP packets can be used to verify that a particular destination is reachable without problems.

Each type of payload packet contains a data field that can be filled by the transmitting machine with data that is intended for a specific type of service. Thus, if the application in use was a virtual terminal session, and the user sent a character by typing on a keyboard, the virtual terminal program in use would create an IP packet intended for the destination machine, which contained a TCP packet intended for the virtual terminal host program on that machine, which contained a data field containing the character typed.

FIG. 3 shows a representation of the process of packet filtering. TCP/IP Packets (hereafter referred to as simply packets) are transmitted through a network by means of devices called routers. Routers are connected to telecommunications circuits or other networks by means of interfaces. A router receives a packet on a particular interface, and by examining that packet and an internal map showing the networks reachable by all of its interfaces (known as a routing table) selects an output interface for that packet and transmits it through that interface. This process is known as routing. In some cases, a router receives a packet which is intended for a destination that cannot be reached from that router. In this case, transmission fails, and the packet is erased and an error message, typically in the form of an ICMP packet (see FIG. 2) is returned to the source of the packet. This type of failure usually indicates a problem with the network. However, some types of routers implement a very similar mechanism for causing packets to fail to reach their destination because of security reasons, not network failure. This process is known as packet filtering, because it allows a network manager to filter out unwanted or unauthorized packets from reaching machines on that network.

Packet-filtering is based on the fields used within a packet (see FIG. 2). A list of rules is loaded into the router that describes the test that a router should apply to each packet to determine whether or not it should be passed or prevented from passing (blocked). This list of rules is known as a Router Packet-Filtering Policy. Generally, this policy is loaded when the router is first turned on, but in practice it may be updated occasionally by human intervention without turning the router off. The packet-filtering policy is a list of

conditions that describe how packets are to be tested for passage through the router. Each entry in the list consists of a condition and an action. If the packet meets the condition, the action is performed. Generally the actions are to pass or to block. For passing, the packet is sent through the router toward its eventual destination. For blocking, the packet is erased and no transmission toward the intended destination takes place. If the packet does not meet the condition, then the next rule in the list is tried, and this process continues until all the rules have been applied or until a condition is met (See FIG. 3*a*).

Packet-filtering routers typically have an implicit final rule which is either ALL PASS or ALL BLOCK, where the condition ALL is one that every packet meets. This final rule, whether implicit or explicit, determines the character of how all of the previous rules must be written. Because of this final condition, there is always a rule that is applicable to a given packet. The case where the final rule is ALL PASS is known as a permissive policy. With ALL PASS, the previous rules must be written so that all cases of packet blockage are specified in advance, and any packet that does not match one of these cases is allowed to pass. The case where the final rule is ALL BLOCK is known as a restrictive policy. With ALL BLOCK, the previous rules must be written so that all cases of a packet passage are specified in advance and any packet that does not match one of these cases is blocked. The conditions for packet matching are based on the fields within the IP, TCP, UDP, and ICMP packets.

FIG. 3 shows two cases of packets entering a router from a source network, intended for a destination network. A Router Packet Filtering Policy has been previously defined for that router. Packet A is the first test case. Packet A has an arbitrary field (designated by the term field) equal to value x. Packet B is the second test case. Packet B has field equal to value y. Packet A enters the router, and the first rule of the packet filtering policy is applied. Because field is equal to x, this condition matches. The action for this rule is "block", so this packet is not permitted to pass through the router. No further rules are applied. Packet B enters the router, and the first rule of the packet filtering policy is applied. Because field is not equal to x in packet B, this rule does not apply. The second rule is applied, and it matches. The action for this rule is "pass" and so this packet is passed on to the destination network. No further rules are applied. Packet filtering rules can be written for varying combinations of conditions and fields.

FIG. 4 shows the points within Device FW at which a Packet-Filtering Policy can be established, labelled there as:

Per-in
Per-esn
Per-psn
Pmr-esn
Pmr-mn
Pmr-psn
Pmr-hsn
Pclr-psn
Pclr-hsn
Pclr-ctc
Pcsr-ctc
Pcsr-psn
Pcsr-cpn

Each of these packet filtering policy application points is located at the interface between a router and a network interface or at the interface between a router and a telecommunications link interface. Each packet filtering policy application point can be configured to permit or deny packet flow between the router and the interface. Policy can be applied to packet flow between the interface and the router or between the router and the interface. These policies may be asymmetric, allowing packet flow in one direction but blocking it in the converse direction, or they may be symmetric, allowing or blocking packet flow in both directions.

FIG. 5 shows an example of an applications level filter. Examples of applications would include remote terminal service, E-mail, file transfer, games. An application filter is a program running on a computer that is interposed in the network between the source and destination networks. This computer acts as a barrier that enforces a security policy between the two networks. A person wishing to use a service on the other side of the barrier must instead connect to the barrier computer. This computer may be configured to run software known as a proxy server. A proxy acts as a surrogate for the service on the far side. A proxy server must be running on the barrier computer for each service that is to be passed through the barrier. The barrier computer must be configured to pass no other services than those specifically allowed by the proxy servers running.

A barrier computer is configured with an Application Filtering Policy which determines the services which will pass and the manner in which they are passed. Services may be passed directly through the proxy or additional constraints may be placed upon them before passage is permitted. A typical constraint is that of authentication. A security policy may be defined that requires that only specific people may be permitted to access services through a barrier computer. In order to determine that an authorized person is connect to the proxy server, the proxy server may require some means of authentication, such as a user name and password. Once the proper credentials are supplied, the connection passes through the proxy server and the barrier, to the service on the other side. Unlike a packet filtering policy, an applications filtering policy is always restrictive, which means that unless a service has an explicitly configured proxy server running on the barrier computer, it will not be passed through.

FIG. 5 shows a typical barrier computer configuration, with an authentication server and three proxy servers running. An applications filtering policy has been configured into the barrier computer, allowing connections to pass on port **23** (typically the remote terminal session port), port **25** (electronic mail) and port **666** (a game). A connection request for port **23**/tcp service arrives at the barrier computer and is passed to the port **23**/tcp proxy. This proxy has a policy of authentication before passage, so the user's credentials are referred to the authentication server, which approves them. The proxy then passes the connection through to the destination machine. A connection request for port **25**/tcp service arrives at the barrier computer and is passed to the port **25**/tcp proxy. This proxy is configured to pass the traffic along, and does so with no further authentication. A packet for port **666**/UDP service arrives at the barrier computer and is passed to the port **666**/udp proxy. This proxy server is configured to pass the packet along and does so with no further authentication. A connection request for port **512**/tcp service, the remote command execution service arrives at the proxy. The application filtering policy does not explicitly cover this service, and so no proxy is configured for this. The request for service is denied.

FIG. 6 shows several points at which an Application Filtering Policy can be established, which are labelled:

Ahes1-esn
Ahes2-esn
Ahcs-hsn
Ahes1-hsn
Ahbs-hsn
Ahes2-pn
Ahbs-pn
Ahag-hsn
Ahas-hsn
Ahag-psn
Als-mn
Amw-mn

Each application filtering policy application point is located at the interface between a computer and a network. Applications filtering policy can be defined based on many different types of criterion, which could include:

Source address
Destination address
Source port
Destination port
Protocol (tcp, UDP, ICMP)
Time of day
User authentication
Server workload

The more constraining criterion an application filtering policy can apply, the more secure it will be. However, a policy configuration for a given customer requires a balance between usability and security if it is to be practical.

FIG. 7 shows the path that traffic would take between a workstation located on the Customer Protected Network (CPN) and needing to access a server on the Internet via a directly-routed, packet-filtered path. In order to achieve this, packet-filtering policy must be set at the following points:

Pcsr-cpn
Pcsr-ctc
Pclr-ctc
Pclr-psn
Per-psn
Per-in

FIG. 8 shows the path that traffic would take for a connection from a workstation on the CPN to access an Internet server by means of an application-filtered connection. In order to achieve this, packet-filtering policy must be set at the following points:

Pcsr-cpn
Pcsr-ctc
Pclr-ctc
Pclr-psn
Per-psn
Per-in

Additionally, application filtering policy must be set at the point labelled "Ahag-psn". The packet filtering policy in this case would be different than that shown in FIG. 7. In particular, the policy set at Pclr-psn would be set to permit traffic only to the HAG machine, and the policy at Per-psn would be set to accept traffic only from the HAG machine.

FIG. 9 shows the path that traffic would take for a connection from a workstation on the Internet to access information on a server on the CPN by means of a directly-routed, packet-filtered connection. In order to achieve this, packet-filtering policy must be set at the following points:

Per-in
Per-psn
Pclr-psn
Pclr-ctc
Pcsr-ctc
Pcsr-cpn

FIG. 10 shows the path that traffic would take for a connection from a workstation on the Internet to access information on a server on the CPN by means of an application-filtered connection. In order to achieve this, packet-filtering policy must be set at the following points:

Per-in
Per-psn
Pclr-psn
Pclr-ctc
Pcsr-ctc
Pcsr-cpn

Additionally, application-filtering policy must be set at the point labelled "Ahag-psn". The packet filtering policy in this case would be different than that shown in FIG. 9. In particular, the policy set at Pclr-psn would be set to permit traffic only to the HAG machine, and the policy at Per-psn would be set to accept traffic only from the HAG machine.

FIG. 11 shows the path that traffic would take for a connection from a workstation on the Internet to access information on a server on the Customer Exported Network (CEN) by means of a directly routed, packet-filtered connection. This is a more typical configuration than the one shown in FIG. 10, because a CEN is specifically set up to support this type of access, while restricting the path from the Internet to the CPN. This configuration also provides much more security for the CES, the CEN, and the CPN. In order to achieve this, packet-filtering policy must be set at the following points:

Per-in
Per-psn
Pclr-psn
Pclr-ctc
Pcsr-ctc
Pcsr-psn

FIG. 12 shows the path that traffic would take for a connection from a workstation on the Internet to access information on a server on the Customer Exported Network (CEN) by means of an application-filtered connection. In order to achieve this, packet-filtering policy must be set at the following points:

Per-in
Per-psn
Pclr-psn
Pclr-ctc
Pcsr-ctc
Pcsr-cen

Additionally, application-filtering policy must be set at the points labelled "Ahag-hsn" and "Ahag-psn". The packet filtering policy in this case would be different than that shown in FIG. 11. In particular, the policy set at Pclr-psn would be set to permit traffic only to the HAG machine, and the policy at Per-psn would be set to accept traffic only from the HAG machine.

FIG. 13 shows the path that traffic from an Internet workstation would take to access a server that was hosted

within Device FW. This type of hosting would provide much more security than the configurations shown in FIGS. **9**, **10**, **11** and **12**, as well as better performance. In order to achieve this, packet-filtering policy is set at the points labelled "Per-in" and "Per-psn" while application-filtering policy is at the points labelled "Ahes1-esn" and "Ahes1-hsn".

FIG. **14** shows the path that traffic from an Internet workstation would take to access a server that was hosted within Device FW. This type of hosting would provide much more security than the configurations shown in FIGS. **9**, **10**, **11** and **12**, as well as better performance. This case differs from that shown in FIG. **13** in that the Hosted External Server (HES1) requires a connection to a "back-end" server such as a database server. This type of configuration is very common for large World-Wide-Web servers. The server in this case is connected to the Customer Exported Network. In order to achieve this, packet-filtering policy must be set at the following points:

Per-in

Per-psn

Pclr-hsn

Pclr-ctc

Pcsr-ctc

Pcsr-cen

Additionally, application-filtering policy must be set at the points labelled "Ahes1-esn" and "Ahes1-hsn".

FIG. **15** shows the path that traffic from an Internet workstation would take to access a server that was hosted within Device FW. This case differs from that shown in FIG. **13** in that the Hosted External Server (HES1) requires a connection to a "back-end" server such as a database server. This type of configuration is very common for large WorldWide-Web servers. In this case, the server is also hosted within Device FW. In order to achieve this, packet-filtering policy is set at the points labelled "Per-in" and "Per-psn" and application-filtering policy is set at the points labelled "Ahes2-esn", "Ahes2-pn", "Ahbs-pn" and "Ahbs-hsn".

FIG. **16** shows the path that a replication connection would use to copy data from a Customer Backend Server (CBS) to a Hosted Backend Server (HBS). By doing this, in conjunction with the configuration shown in FIG. **15**, provides a much more secure and robust mechanism for building a high-performance, high-capacity External Server. In order to achieve this, packet-filtering policy is set at the points labelled "Per-in" and "Per-psn" and applications filtering policy is set at the points labelled "Ahbs-pn" and "Ahbs-hsn".

FIG. **17** shows the path of traffic from a workstation on the CPN for directly-routed, packet-filtered access to a Hosted Customer Server. This approach would be taken for servers that would normally reside on the CPN, but for various reasons such as security or difficulty of administration, have been outsourced and now reside within Device FW. In order to accomplish this, packet-filtering policy must be set at the following points:

Pcsr-cpn

Pcsr-ctc

Pclr-ctc

Pclr-hsn

Additionally, applications filtering policy must be set at the point labelled "Ahes-hsn".

It should be readily apparent to those skilled in the art that, after reading this description, a wide range of modifications may be made to the security policy of a particular embodiment of this apparatus. The scope of the invention should, therefore, be determined not with reference to the above description, but instead should be determined with reference to the appended claims along with their full scope of equivalents.

What is claimed is:

**1**. A security handler for packet transfer between an insecure network and a secure network wherein packets are passed or blocked between the insecure network and the secure network to secure the secure network against attacks from the insecure network, the security handler comprising:

means for obtaining customer security policies, wherein a customer security policy is a set of one or more rule defining a set of capabilities that are allowed or disallowed for a given customer's secure network, modifiable in response to security attacks encountered, and wherein customer security policies can be distinct for distinct customers;

a plurality of packet processing components;

a plurality of communication paths between components of the plurality of packet processing components; and

configurable policy enforcement means, at each connection of a communication path and a packet processing component, for enforcing a packet policy for packets transported between the communication path and the packet processing component, wherein the packet policy is a function of the customer security policies.

**2**. A method of securing a plurality of secure customer networks while connected to one or more insecure networks, wherein packet traffic on an insecure network can be generated and observed without authorization, comprising the steps of:

interposing a packet processor between at least one of the secure networks and at least one of the insecure networks, wherein the packet processor includes a plurality of paths over which packets are transported between the at least one insecure network and the at least one secure network;

identifying control points within the packet processor, wherein a control point is a node through which packet traffic having predetermined characteristics flows and each control point has a set of one or more predetermined characteristics associated therewith;

storing customer security policies for each of the plurality of secure customer networks, wherein a customer security policy specifies capabilities that are allowed or not allowed with respect to an associated secure customer network, wherein a capability is provided by packet traffic having predetermined characteristics associated with that capability and wherein customer security policies can be distinct for distinct customers;

controlling packet traffic flow between control points in accordance with the customer security policies to limit the capabilities of-each secured customer network to those capabilities allowed by the customer security policies; and

modifying customer sercurity policies in response to security attacks encountered.

**3**. The method of claim **2**, wherein the capabilities controlled in the step of controlling packet traffic flow include e-mail transport, hypertext transport protocol packet transport, file transport and remote terminal access.

**4**. The method of claim **2**, wherein the predetermined characteristics include a packet source address and port, a packet destination address and port, a protocol, a time of day, and an authorization status.

5. A method of controlling a data exchange between a customer network and an external network comprising the steps of:

for each data object, a data object comprising one or more packets, identifying object features including a packet source address and port, a packet destination address and port and an object protocol;

applying policy rules from a policy table to features of the data object, where a policy rule is a variable rule, determined by a customer policy, for excluding or including the data object based on the object's features, wherein customer policies can be distinct for distinct customers and customer policies are modifiable in response to security attacks encountered; and

if a policy rule indicates that a data object should be included and no policy rule indicates that the data object would be excluded, forwarding the data object from its source to destination.

6. A common hardware platform for handling multiple customer networks, each of which may have different security policies, the common hardware platform comprising:

a policy database containing a plurality of policies for the multiple customer networks, wherein a policy is a rule about what capabilities are allowed or disallowed for traffic between a customer network and another network when the conditions of the rule are met and wherein policies can be distinct for distinct customers and policies are modifiable in response to security attacks encountered; and

means for controlling data traffic between a customer network and another network based on the policies in the policy database.

7. The common hardware platform of claim 6, wherein the means for controlling data traffic selects an action from permitting the data traffic to flow, denying the data traffic and redirecting the data traffic.

* * * * *

US006341130B1

(12) **United States Patent**
**Lakshman et al.**

(10) **Patent No.: US 6,341,130 B1**
(45) **Date of Patent: Jan. 22, 2002**

(54) **PACKET CLASSIFICATION METHOD AND APPARATUS EMPLOYING TWO FIELDS**

(75) Inventors: **Tirunell V. Lakshman**, Morganville; **Dimitrios Stiliadis**, Matawan, both of NJ (US)

(73) Assignee: **Lucent Technologies, Inc.**, Murray Hill, NJ (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/146,122**

(22) Filed: **Sep. 2, 1998**

**Related U.S. Application Data**

(60) Provisional application No. 60/073,996, filed on Feb. 9, 1998.

(51) **Int. Cl.**[7] ..... **H04L 12/66**

(52) **U.S. Cl.** ..... **370/389**; 370/351; 370/392; 370/401; 709/238; 709/245

(58) **Field of Search** ..... 370/389, 390, 370/400, 401, 402, 403, 404, 405, 406, 407, 408, 410, 466, 467, 351, 428, 392; 709/238, 230, 246, 249, 235, 245

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | |
|---|---|---|---|---|
| 5,781,772 A | * | 7/1998 | Wilkinson, III et al. | 370/401 |
| 5,951,651 A | * | 9/1999 | Lakshman et al. | 370/389 |
| 5,995,971 A | * | 11/1999 | Douceur et al. | 370/408 |
| 6,147,976 A | * | 11/2000 | Shand et al. | 370/392 |

OTHER PUBLICATIONS

Bailey et al., PATHFINDER: A Pattern–Based Packet Classifier, University of Arizona, pp. 1–9, 1994.*

McCanne et al, The BSD Packet Filter: A New Architecture for User–level Packet Capture, Lawrence Berkley Laboratory, pp. 1–11, 1992.*

Miei et al., Parallelization of IP–Packet Filter Rules, Nippon Telegraph and Telephone Co., pp. 381–388, 1997.*

DeBerg et al., Two–and Three–Dimensional Point Location in Rectangular Subdivision, University of British Columbia, pp. 1–17, 1995.*

Alessandri, Access Control List Processing In Hardware, Diploma Thesis, ETH, pp. 1–85, 1997.*

* cited by examiner

*Primary Examiner*—Wellington Chin
*Assistant Examiner*—Frank Duong
(74) *Attorney, Agent, or Firm*—Steve Mendelsohn; Ian M. Hughes

(57) **ABSTRACT**

A packet filter for a router performs generalized packet filtering allowing range matches in two dimensions, where ranges in one dimension at least one dimension is defined as a power of two. To associate a filter rule with a received packet EP, the packet filter employs a 2-dimensional interval search and memory look-up with the filter-rule table. Values of $s_m$ of filter-rule $r_m=(s_m,d_m)$ in one dimension are desirably ranges that are a power of two, such as prefix ranges, which are represented by a binary value having a "length" defined as the number of bits to of the prefix. The $d_m$ may be single points, ranges defined as prefix ranges, and/or ranges defined as continuous ranges. The packet filter employs preprocessing of the filter-rules based on prefix length as a power of 2 in one dimension and decomposition of overlapping segments into non-overlapping intervals in the other dimension to form the filter-rule table. A preprocessing algorithm searches in one dimension through filter rules and arranges the corresponding filter-rule rectangle segments according to prefix length. Then, in the other dimension, the overlapping filter rectangle segments are decomposed into non-overlapping intervals, and the highest priority filter-rule overlapping each non-overlapping interval is associated with that interval. A filter-rule table is then constructed with entries ordered according to prefix length and non-overlapping interval, each entry associated with a particular filter-rule. A packet classification algorithm then matches the field or other parameter information in the packet to the filter-rule table entries to identify the filter-rule rectangle associated with the filter-rule to be applied to the packet.

**30 Claims, 9 Drawing Sheets**

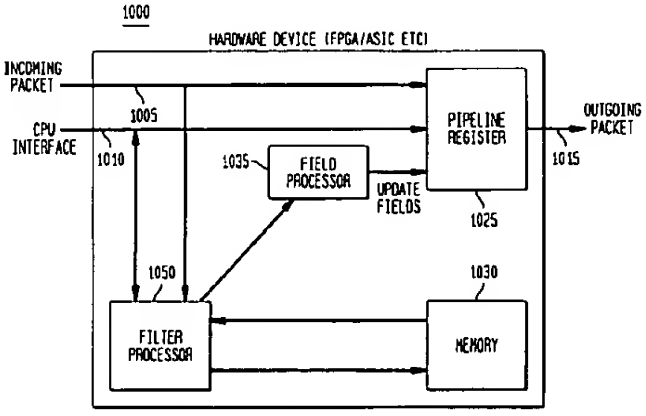


FIG. 1
(PRIOR ART)

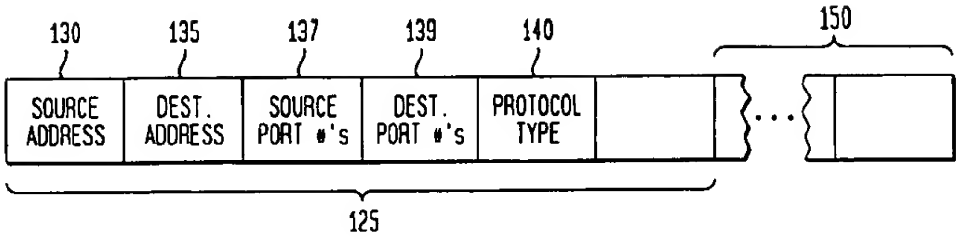



FIG. 2
(PRIOR ART)

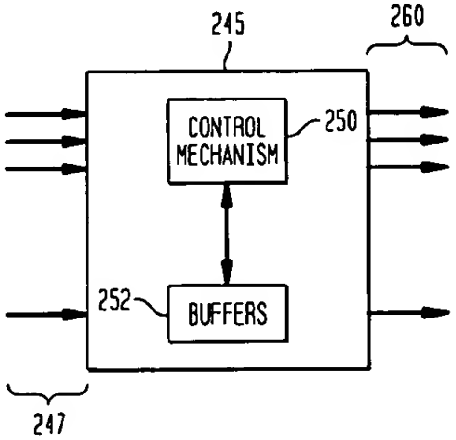



FIG. 3

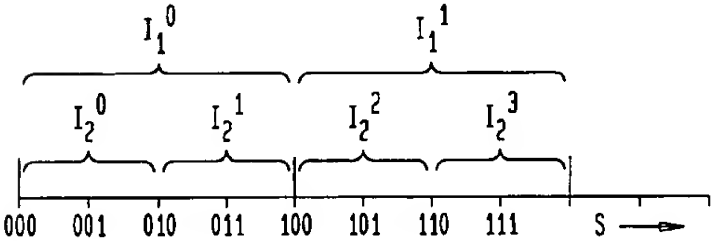



FIG. 4

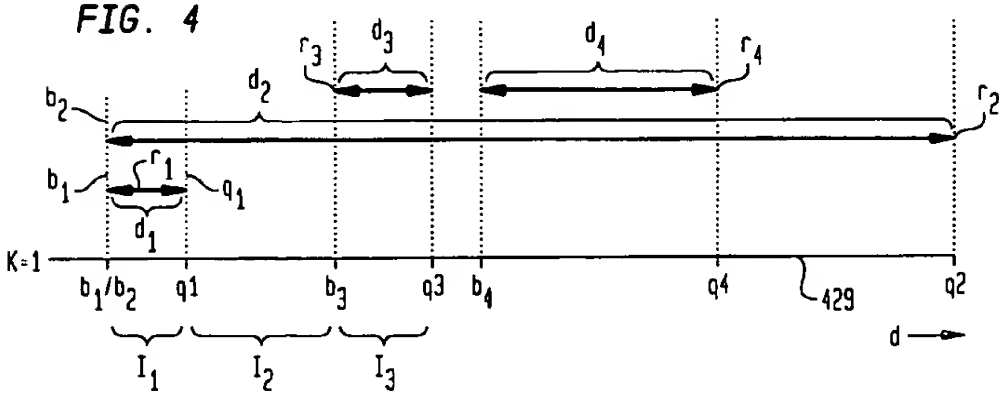


FIG. 5

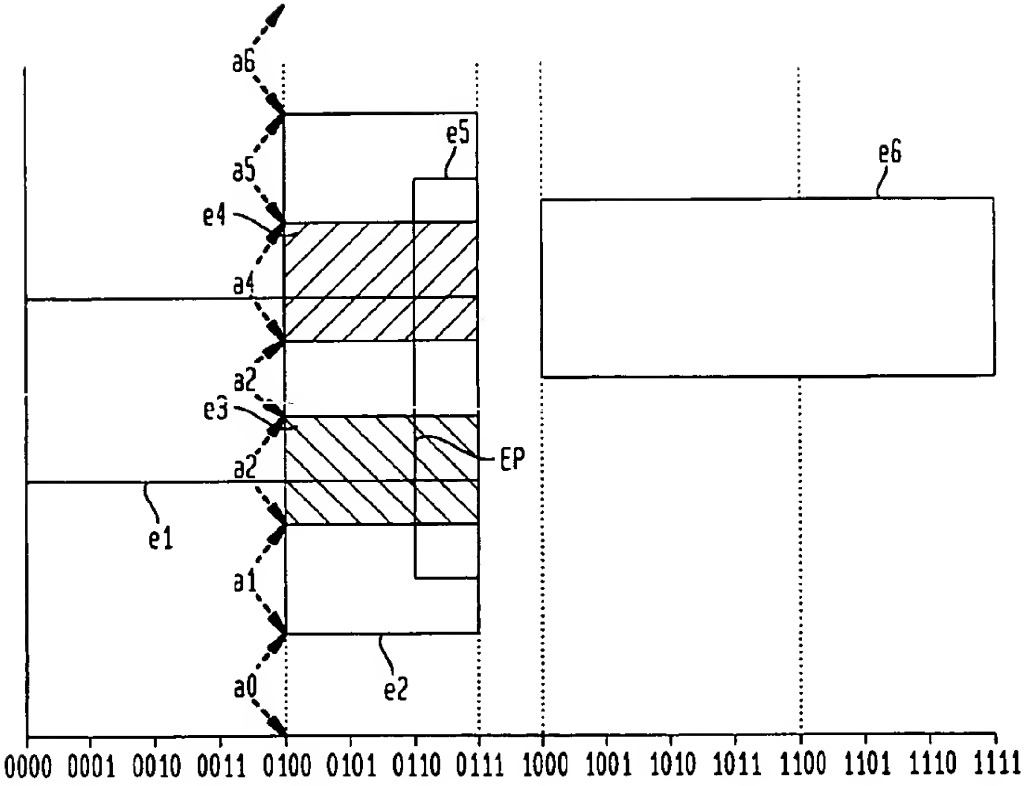

a6
a5
e4
a4
a2
e3
a2
e1
a1
a0
e2
e5
EP
e6
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

FIG. 6

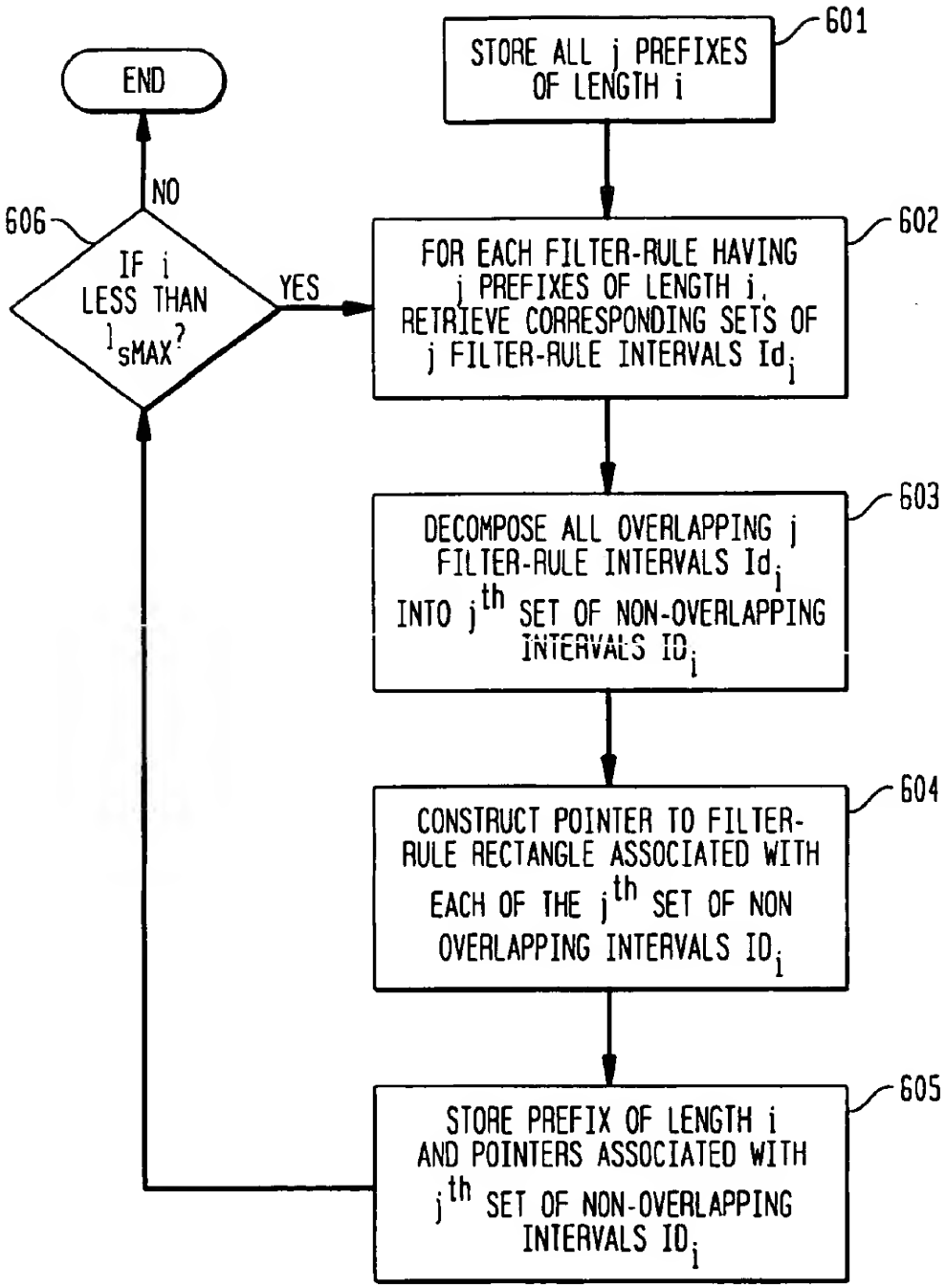

601
STORE ALL j PREFIXES OF LENGTH i
602
FOR EACH FILTER-RULE HAVING j PREFIXES OF LENGTH i, RETRIEVE CORRESPONDING SETS OF j FILTER-RULE INTERVALS Id_i
603
DECOMPOSE ALL OVERLAPPING j FILTER-RULE INTERVALS Id_i INTO j^th SET OF NON-OVERLAPPING INTERVALS ID_i
604
CONSTRUCT POINTER TO FILTER-RULE RECTANGLE ASSOCIATED WITH EACH OF THE j^th SET OF NON OVERLAPPING INTERVALS ID_i
605
STORE PREFIX OF LENGTH i AND POINTERS ASSOCIATED WITH j^th SET OF NON-OVERLAPPING INTERVALS ID_i
606
IF i LESS THAN l_sMAX?
YES
NO
END

FIG. 7

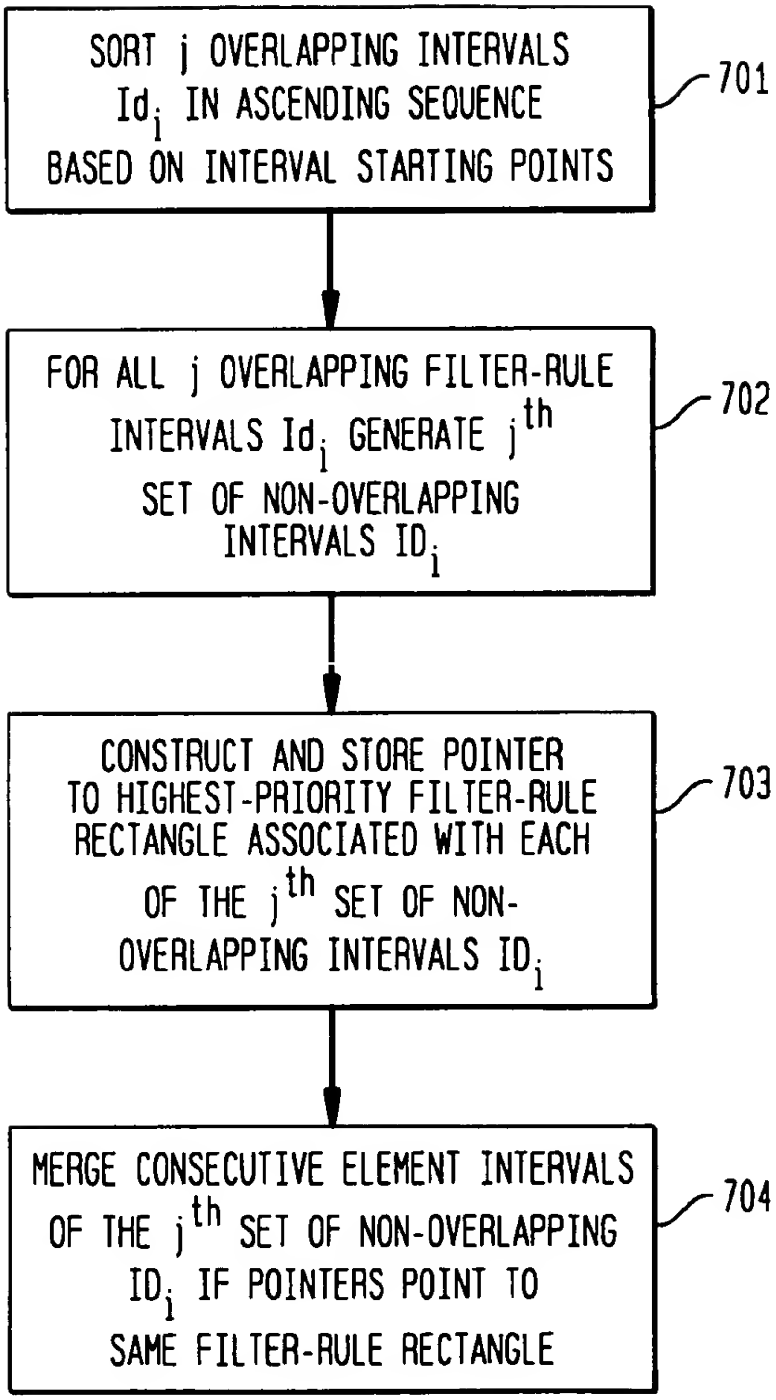
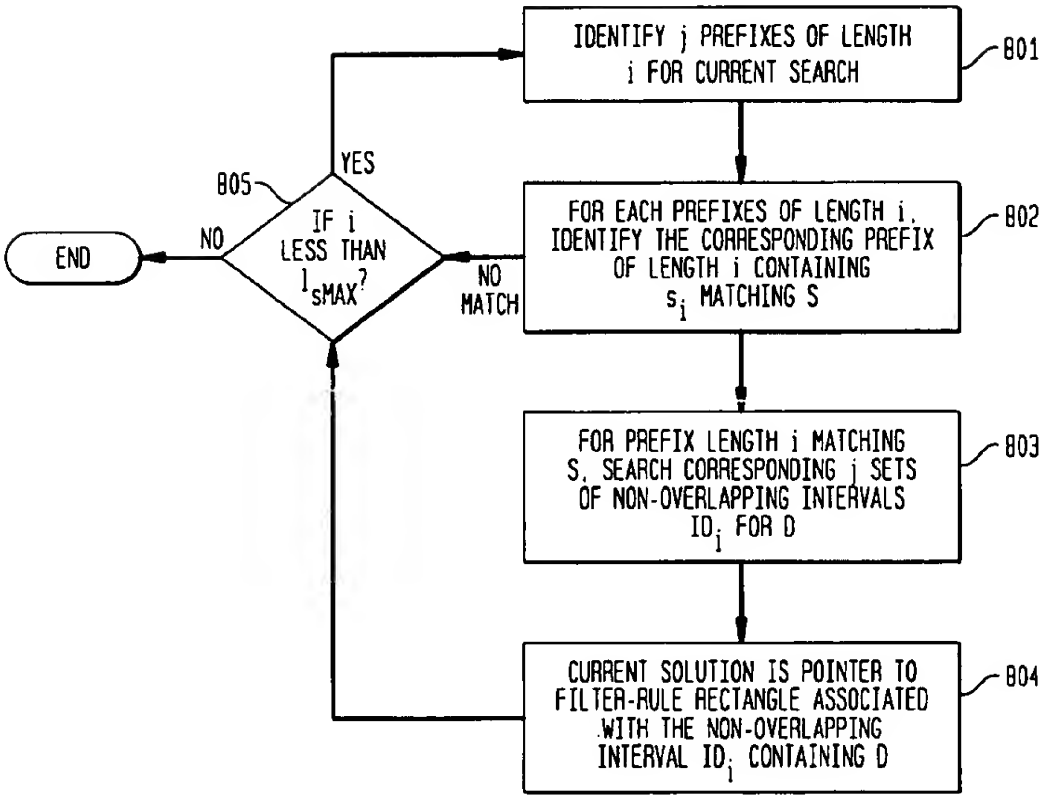

SORT j OVERLAPPING INTERVALS Id$_i$ IN ASCENDING SEQUENCE BASED ON INTERVAL STARTING POINTS
701
FOR ALL j OVERLAPPING FILTER-RULE INTERVALS Id$_i$ GENERATE j$^{th}$ SET OF NON-OVERLAPPING INTERVALS ID$_i$
702
CONSTRUCT AND STORE POINTER TO HIGHEST-PRIORITY FILTER-RULE RECTANGLE ASSOCIATED WITH EACH OF THE j$^{th}$ SET OF NON-OVERLAPPING INTERVALS ID$_i$
703
MERGE CONSECUTIVE ELEMENT INTERVALS OF THE j$^{th}$ SET OF NON-OVERLAPPING ID$_i$ IF POINTERS POINT TO SAME FILTER-RULE RECTANGLE
704

FIG. 8


IDENTIFY j PREFIXES OF LENGTH i FOR CURRENT SEARCH
801
FOR EACH PREFIXES OF LENGTH i, IDENTIFY THE CORRESPONDING PREFIX OF LENGTH i CONTAINING $s_i$ MATCHING S
802
FOR PREFIX LENGTH i MATCHING S, SEARCH CORRESPONDING j SETS OF NON-OVERLAPPING INTERVALS $ID_j$ FOR D
803
CURRENT SOLUTION IS POINTER TO FILTER-RULE RECTANGLE ASSOCIATED WITH THE NON-OVERLAPPING INTERVAL $ID_j$ CONTAINING D
804
805
IF i LESS THAN $l_{sMAX}$?
YES
NO
NO MATCH
END

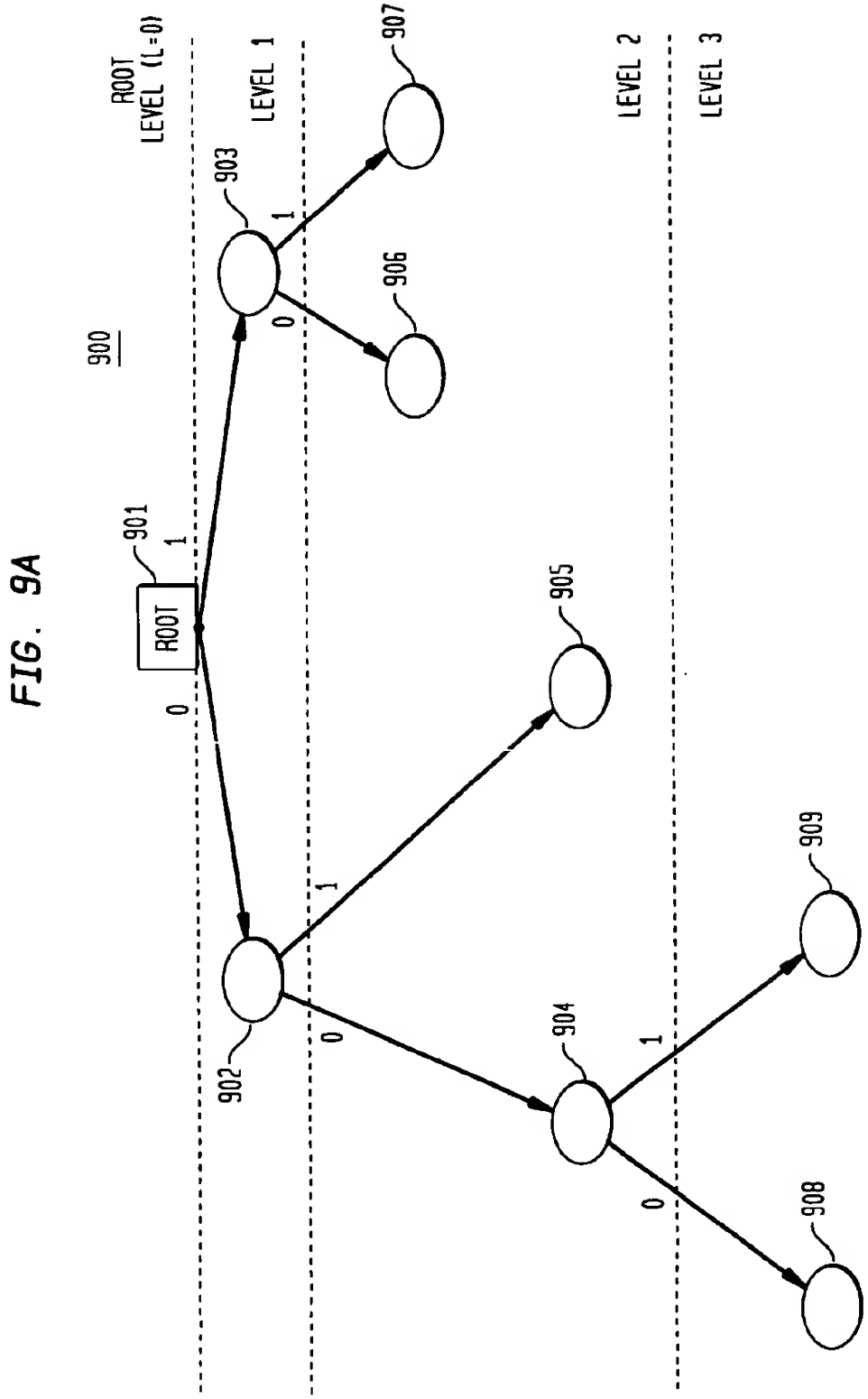

ROOT
LEVEL (L=0)
LEVEL 1
LEVEL 2
LEVEL 3
900
901
ROOT
903
906
907
905
902
904
908
909
1
0

FIG. 9A

FIG. 9B

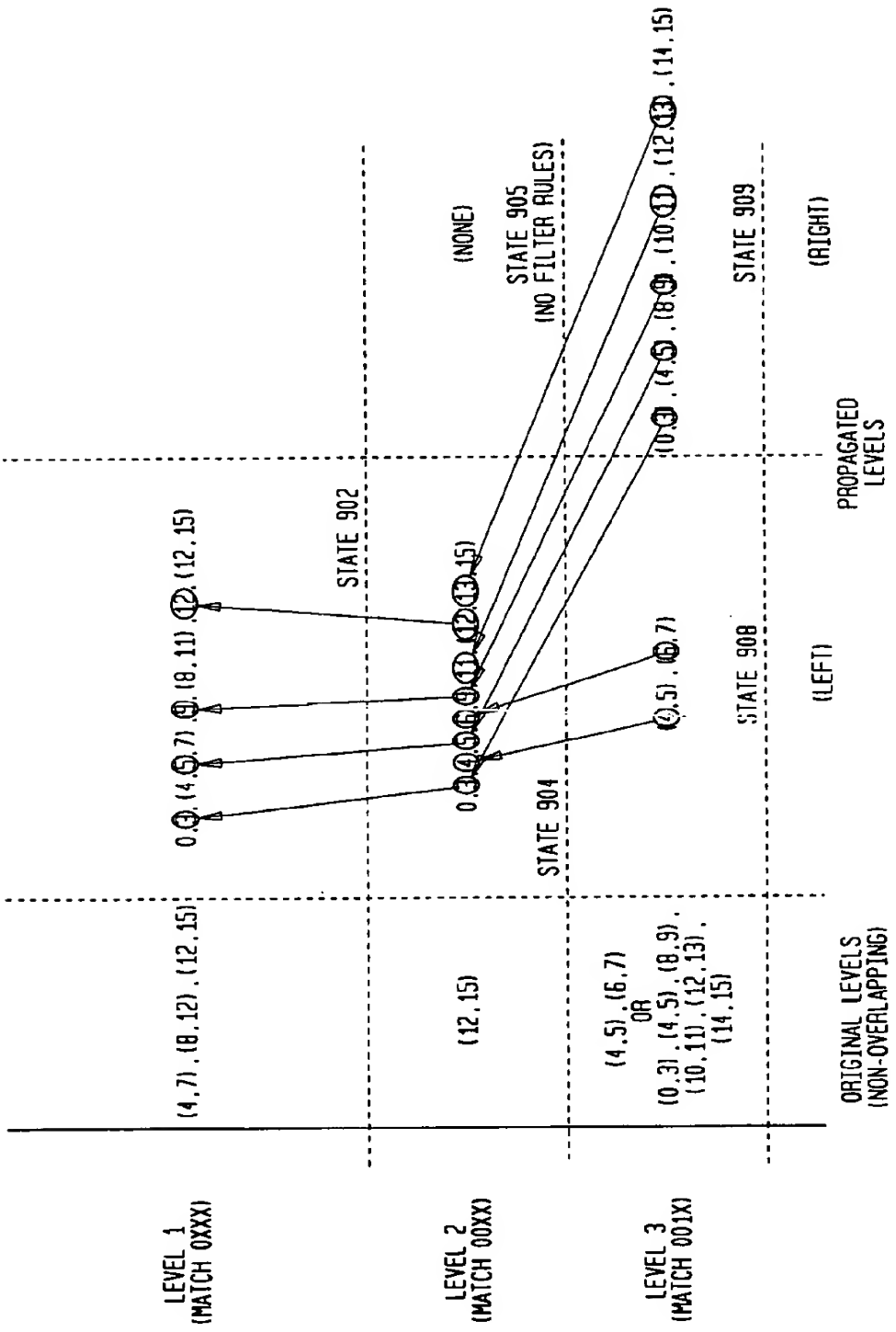

LEVEL 1
(MATCH 0XXX)
LEVEL 2
(MATCH 00XX)
LEVEL 3
(MATCH 001X)
ORIGINAL LEVELS
(NON-OVERLAPPING)
PROPAGATED
LEVELS
(LEFT)
(NONE)
(RIGHT)
STATE 902
STATE 904
STATE 905
(NO FILTER RULES)
STATE 908
STATE 909
(4,7),(8,12),(12,15)
(12,15)
(4,5),(6,7)
OR
(0,3),(4,5),(8,9),
(10,11),(12,13),
(14,15)

FIG. 10

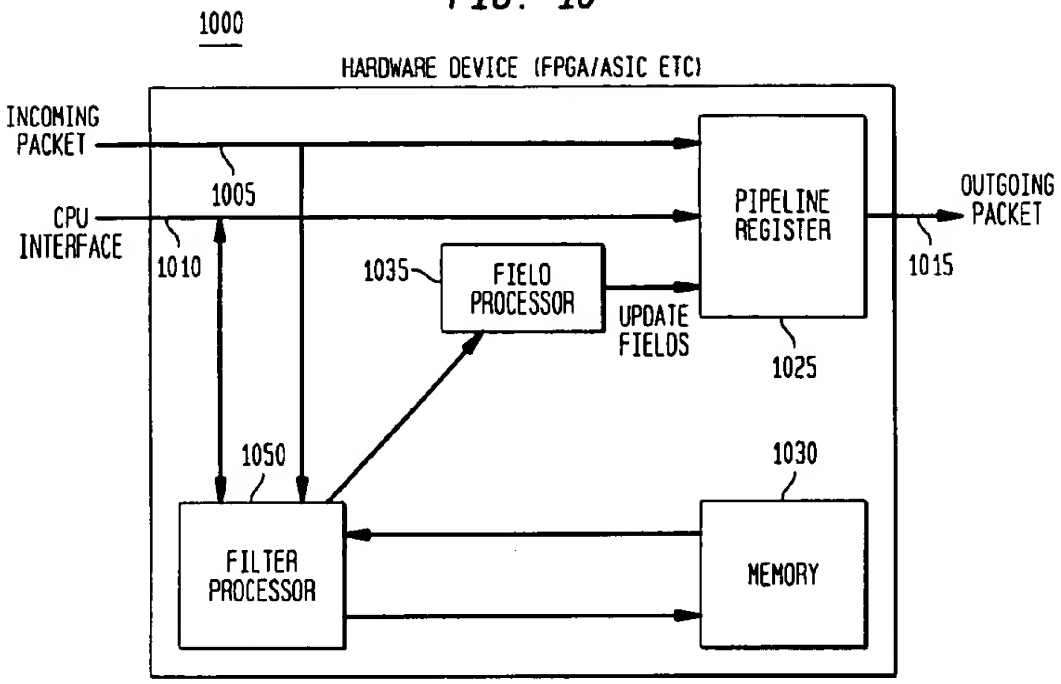

1000
HARDWARE DEVICE (FPGA/ASIC ETC)
INCOMING PACKET
1005
CPU INTERFACE
1010
1035
FIELD PROCESSOR
UPDATE FIELDS
PIPELINE REGISTER
1025
OUTGOING PACKET
1015
1050
1030
FILTER PROCESSOR
MEMORY

FIG. 11

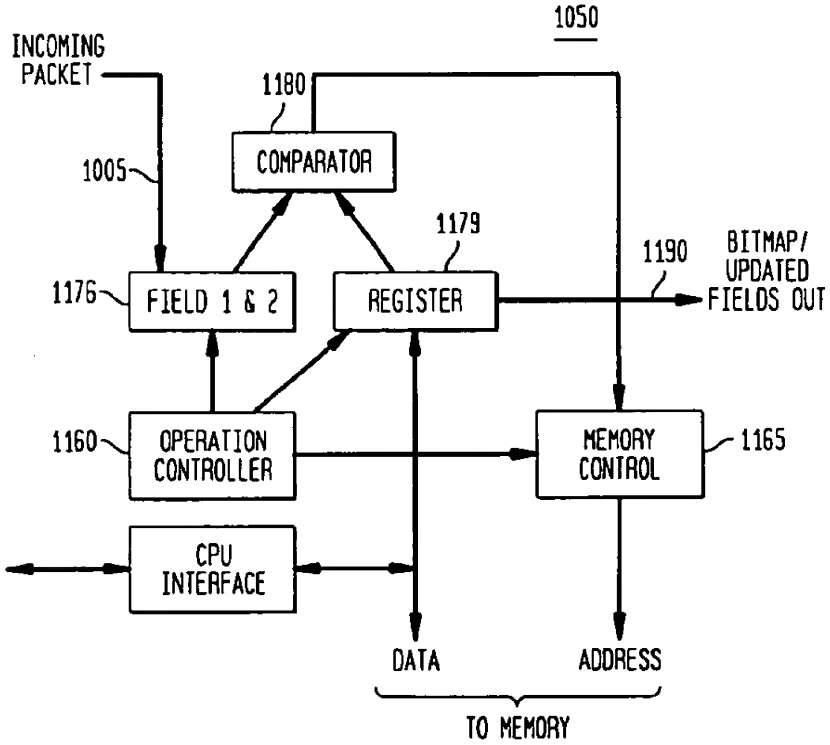
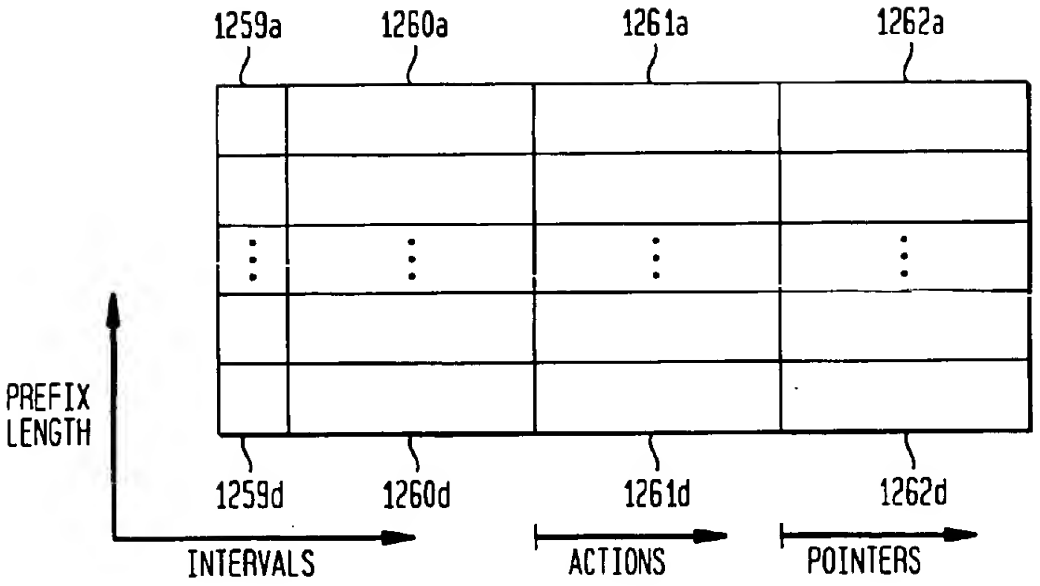

1050
INCOMING PACKET
1180
COMPARATOR
1005
1179
1190
BITMAP/ UPDATED FIELDS OUT
1176
FIELD 1 & 2
REGISTER
1160
OPERATION CONTROLLER
MEMORY CONTROL
1165
CPU INTERFACE
DATA
ADDRESS
TO MEMORY

FIG. 12


1259a
1260a
1261a
1262a
PREFIX LENGTH
1259d
1260d
1261d
1262d
INTERVALS
ACTIONS
POINTERS

# PACKET CLASSIFICATION METHOD AND APPARATUS EMPLOYING TWO FIELDS

## CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of the filing date of U.S. provisional application No. 60/073,996, filed on Feb. 9, 1998.

## BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to packet forwarding engines used in telecommunications, and, in particular, to router algorithms and architectures for supporting packet filter operations using two packet fields.

2. Description of the Related Art

Packet-based communication networks, such as the Internet, typically employ a known protocol over paths or links through the network. Commonly known protocols are, for example, Transmission Control Protocol/Internet Protocol (TCP/IP) or Reservation Set-up Protocol (RSVP). Routers provided in a communication network provide a packet forwarding function whereby input data, usually in the form of one or more data packets, is switched or routed to a further destination along a network link. FIG. **1** shows a typical form of a data packet **20**, which may be of variable length. Data packet **20** comprises, for example, a header **125** and payload data **150**. Header **125** contains fields or parameters, such as a source address **130** where the data originates and at least one destination address **135** where the data is to be routed. Another parameter in the header **125** may be a protocol type **140** identifying a particular protocol employed in the communication network.

FIG. **2** shows a router **245** of a network node receiving streams or flows of data packets from input links **247** and routing these packet streams or flows to output links **260**. To perform a forwarding function, router **245** receives a data packet at an input link **247** and a control mechanism **250** within the router utilizes an independently generated look-up table (not shown) to determine to which output link **260** the packet should be routed. It is understood that the packet may first be queued in buffers **252** before being routed, and that the forwarding function is desirably performed at a high rate for high forwarding throughput.

Source and destination addresses may be logical addresses of end hosts (not shown). Thus, data packet **20** of FIG. **1** may further comprise unique source port numbers **137** and destination port numbers **139**. Header **125** may also include, for example, certain types of flags (not shown) in accordance with protocol type **140**, such as TCP, depending upon the receiver or transmitter application.

Network service providers, while using a shared backbone infrastructure, may provide different services to different customers based on different requirements. Such requirements may be different service pricing, security, or Quality of Service (QoS). To provide these differentiated services, routers typically include a mechanism for 1) classifying and isolating traffic, or packet flows, from different customers, 2) preventing unauthorized users from accessing specific parts of the network, and 3) providing customized performance and bandwidth in accordance with customer expectations and pricing.

Consequently, in addition to the packet forwarding function, router **245** of FIG. **2** may perform a packet filtering function. Packet filtering may be employed, for example, as "firewall protection" to prevent data or other information from being routed to certain specified destinations within the network. To perform packet filtering, the router **245** may be provided with a table or list of filter rules specifying that routing of packets sent from one or more of specified sources is denied or that specific action is to be taken for that packet having a specified source address. Such packet filtering may be employed by layer four switching applications.

Specifically, packet-filtering parses fields from the packet header **125** including, for example, both the source and destination addresses. Parsing allows each incoming packet to be classified using filter rules defined by network management software, routing protocols, or real-time reservation protocols such as RSVP.

Filter rules may also specify, for example, that received packets with fields specifying that a particular destination address should or should not be forwarded through specific output links, or that some other specific action should be taken before routing such received packets. Thus, a variety of filter rules may be implemented based on packet field information. For example, such filter rules might be based on 1) source addresses; 2) destination addresses; 3) source ports; 4) destination ports; and/or 5) any combination of these fields.

Packet filtering of the prior art generally requires either an exact match operation of the fields or a match operation defined in terms of field ranges for a filter rule. Field ranges may specify, for example, ranges of source addresses, destination addresses, source/destination port numbers, and/or protocol types. Filter rules are then applied to every packet that the router receives; that is, for each packet received by the router, every filter rule is successively applied to each packet to ascertain whether that packet is to be forwarded, restricted, or re-routed according to the filter rule. However, implementation of a large number of filter rules in a router (e.g. 500 or more) is time consuming with respect to processor execution time since all filter rules must be tested. Hence, routers implementing filters having a large number of filter rules have decreased throughput, compromising a quality of service (QoS). Thus, for a router such as router **245** to maintain a relatively high level of throughput, the filtering function must be performed at very high rate.

The IP packet header fields may contain up to 128 bits of parameter information, including source and destination addresses, physical source and destination port numbers, interface number, protocol type, etc. Each of the fields or parameters in the header may be represented as being along an axis of a dimension. The general packet classification problem of a packet filter may then be modeled as a point-location in a multi-dimensional space. One or more field values of a packet define a point in the multi-dimensional space. A packet filter rule associated with a range of values of each defines an object in the multi-dimensional space.

A point-location algorithm in a multi-dimensional space with multi-dimensional objects finds the object that a particular point belongs to. In other words, given a received point $EP=\{E_1, E_2, \ldots E_D\}$ in a space having D dimensions, find one or more of a set of n D-dimensional objects including the point (n being an integer greater then 0). The general case of D>3 dimensions may be considered for the problem of packet classification. As is known in the art, the best algorithms optimized with respect to time or space have either an $O(\log^{D-1} n)$ time complexity with O(n) space or an O(log n) time complexity with $O(n^D)$ space, where O(·) mathematically represents "on the order of." Comparing

algorithms on the basis of the order of operations is particularly useful since operations may be related to memory requirements (space) and execution time (time complexity).

Though algorithms with these complexity bounds are useful in many applications, they are not currently useful for packet filtering. First, packet filtering must complete within a specified amount of time, which generally forces a value for n to be relatively small relative to asymptotic bounds, but routers typically filter packets with a number of filter rules in the range of a few thousand to tens of thousands. Consequently, even point-location algorithms with poly-logarithmic time bounds are not practical for use in a high-speed router.

For example, router **245** desirably processes n=1K filter rules of D=5 dimensions within 1 $\mu$s to sustain a 1 million-packets-per-second throughput. However, an algorithm employed with $O(\log^{D-1} n)$ complexity and O(n) space has a $\log^4 1024$ execution time and O(1024) space, which requires 10K memory accesses (look-ups) per packet. If an O(log n) time $O(n^4)$ space algorithm is employed, then the space requirement becomes prohibitively large (greater than 1000 Gigabytes).

For the special case of two dimensions, the filter rules defined for field ranges are modeled as objects in two dimensions, for example, forming rectangles in the 2-dimensional space. For a 2-dimensional space having non-overlapping rectangles, some packet filter algorithms have logarithmic complexity and near-linear space complexity. However, these algorithms do not consider the special problem related to arbitrary overlapping rectangles in the multi-dimensional space requiring a decision of which overlapping filter rules to apply to a packet. The problem may be resolved through a priority of the longest field prefix. An algorithm of the prior art where the time complexity is O(log(log N)) is based on stratified tree searches in a finite space of discrete values. Examples of these algorithms are discussed in, for example, M. De Berg, M. van Kreveld, and J. Snoeyink, Two- and Three-dimensional Point Location in Rectangular Subdivisions, Journal of Algorithms, 18:256–277, 1995. Data structures employed by this prior art algorithm require a perfect hashing operation in every level of the tree. The pre-processing complexity, without using a randomized algorithm, of calculating the perfect hash is $O(\min(hV,n^3)$, where h is the number of hash functions that must be calculated and V is the size of the space. Consequently, for a 2-dimensional space, longest-prefix lookups may result in executions requiring $2^{32}$ cycles, even for a relatively small number of filter rules, even if pre-processing is only required once every several seconds.

## SUMMARY OF THE INVENTION

The present invention relates to a packet filter associating at least one filter rule with a packet, each filter rule and the packet characterized by values in first and second dimensions, the filter rule to be applied to the packet by a router in a communications network. In accordance with an exemplary embodiment, a filter-rule table is provided with each entry of the filter-rule table corresponding to a prefix value having a length in the first dimension and at least one interval in the second dimension. Each prefix value matching the value of the packet in the first dimension is identified, and each interval corresponding to identified prefix values containing the value of the packet in the second dimension is retrieved. A solution interval is determined as the interval associated with the prefix value associated with a predetermined metric and containing the value of the packet in the second dimension; and the filter rule corresponding to the solution interval is associated with the packet.

In accordance with another exemplary embodiment, the filter-rule table is created by first assigning each filter-rule to one or more prefix values based on the values in the first dimension; and then projecting, for each prefix value having the same length, values of each corresponding filter rule of the prefix value onto the second dimension to define at least one filter-rule segment. Each filter-rule segment is decomposed into one or more non-overlapping intervals associated with each prefix value having the same length and corresponding filter rule in the second dimension; and a pointer is generated for each non-overlapping interval identifying each filter rule contained in the non-overlapping interval. The pointer is stored as an entry of the filter-rule table associated with a prefix value length and a non-overlapping interval.

## BRIEF DESCRIPTION OF THE DRAWINGS

Other aspects, features, and advantages of the present invention will become more fully apparent from the following detailed description, the appended claims, and the accompanying drawings in which:

FIG. **1** shows a typical form of a data packet of a communications network;

FIG. **2** shows a router of a network node receiving and forwarding packet streams;

FIG. **3** illustratively depicts prefix ranges of a field in an s-dimension where the prefix ranges are a power of two;

FIG. **4** illustratively depicts segments of a filter rule having one or more field ranges of destination addresses projected as horizontal intervals;

FIG. **5** illustrates a 2-dimensional space for an exemplary packet filter in accordance with the first embodiment of the present invention;

FIG. **6** illustrate steps of an exemplary pre-processing algorithm in accordance with the present invention;

FIG. **7** illustrate steps of decomposing overlapping intervals into non-overlapping intervals as shown in FIG. **6**;

FIG. **8** illustrates steps of an exemplary classification algorithm in accordance with the present invention;

FIG. **9A** illustrates an example of trie structure of an exemplary embodiment employing virtual intervals to reduce search time of a classification algorithm;

FIG. **9B** illustrates an example of point propagation of an exemplary embodiment employing virtual intervals to reduce search time of a classification algorithm;

FIG. **10** illustrates a hardware system for implementation of the packet filter in accordance with the present invention in a packet forwarding engine or router;

FIG. **11** shows a filter processor receiving incoming packets, storing field parameters and classifying a packet in accordance with the present invention; and

FIG. **12** shows an example memory organization of a filter-rule table for the system illustrated in FIG. **10**, which depicts a filter-rule.

## DETAILED DESCRIPTION

For exemplary embodiments of the present invention, a packet filter associates a 2-dimensional filter rule with an arriving packet EP having fields S and D. For a unicast forwarding packet filter, these values S and D may be source and destination address values, respectively, of the packet. For a multicast forwarding packet filter, the value S may be

the source address value of a packet and D a group identifier (ID) that identifies the multicast group that the packet may be forwarded to. The value for S may be contained in a range of binary values s, s being associated with an axis in one dimension (the s-dimension). Similarly, the value for D may be contained in a range of binary values d, d being associated with another axis in another dimension (the d-dimension). The packet filter includes a set of n packet-filtering rules RP having 2 dimensional filter rules $r_1$ through $r_n$ to be associated with the packet. Each filter rule $r_m$, m an integer greater than 0, may be denoted as $r_m=\{s_m,d_m\}$, which is a set of two field ranges $s_m$ and $d_m$ in the s-dimension and d-dimension that define the filter rule $r_m$ in the 2-dimensional space.

To associate a filter rule with a received packet EP, the packet filter employs a 2-dimensional interval search and memory look-up with the filter-rule table. Locating a pair of values S and D for fields of a packet EP and associating a 2-dimensional filter rule with the packet may be modeled as a point-location problem in a 2-dimensional space. The packet EP having field values S and D arrives at the router and is defined as a query point (S, D) of a 2-dimensional space. For the point-location problem where packet filtering involves orthogonal rectangular ranges, a search in 2-dimensions of a 2-dimensional, orthogonal, rectangular range decomposes each rectangle into a set of 1-dimensional filter-rule intervals to allow 1-dimensional searches over 1-dimensional intervals.

For a simple embodiment, preprocessing of filter-rules may construct the filter-rule table as a 2-dimensional look-up table comprising filter-rule pairs $(s_m,d_m)$, m an integer greater than 0, where each $s_m$ is a prefix of possible source addresses and each $d_m$ is a contiguous range, or a single point, of possible destination addresses or group IDs. For the table, each pair $(s_m,d_m)$ defines a filter-rule rectangle $r_m=\{s_m, d_m\}$ for the n packet-filtering rules $r_1$ through $r_n$ in 2-dimensions, and rectangles may overlap. The point-location-in-a-2-dimensional-space operates as follows: given the query point (S, D) of packet EP, the search or look-up algorithm for packet classification finds an enclosing filter-rule rectangle $r_m=(s_m,d_m)$, if any, such that the query point (S, D) is contained in $r_m$, and such that $s_m$ is the most specific filter according to a predefined metric, such as, for example, the longest matching prefix of field value S or the highest priority rule for a given prefix length.

For Internet Protocol (IP) routers employing an algorithm in accordance with the present invention, look-up tables may have as many as $2^{16}$ entries or more. Also, algorithms employed may generally be evaluated based on worst-case performance since queuing for header processing is desirably avoided to provide a specific Quality of Service (QoS). For the exemplary filter-rule table, a value n may be defined to denote a number of entries in the table, for example a multicast forwarding table, corresponding to the n filter rules $r_1$ through $r_n$. An nxn array may be formed in a memory with each entry representing the highest-priority filter-rule rectangle of the n filter rules $r_1$ through $r_n$ enclosing a point corresponding to the coordinates represented by the entry. An exemplary classification (i.e., look-up) algorithm that employs this simple table may employ two binary searches, one for each of the dimension. This exemplary classification algorithm may require O(log n) time and $O(n^2)$ memory space. The $O(n^2)$ memory space is due to one rectangle being represented in O(n) locations. Such simple table might not be preferred, however, for a high-speed router when the number of filtering rules is $n=2^{16}$ or greater since the required memory space or memory access time may be excessive.

Consequently, preferred embodiments of the present invention employ preprocessing of the filter-rules based on prefix length as a power of 2 in one dimension and decomposition of overlapping segments into non-overlapping intervals in the other dimension to form the filter-rule table. A packet filter of the present invention first searches in one dimension through filter rules and arranges the corresponding filter-rule rectangle segments according to prefix length. Then, in the other dimension, the overlapping filter rectangle segments are decomposed into non-overlapping intervals, and the highest priority filter-rule overlapping each non-overlapping interval is associated with that interval. A filter-rule table is then constructed with entries ordered according to prefix length and non-overlapping interval, each entry associated with a particular filter-rule. This filter-rule table is constructed within a router prior to processing of received packets. Packet classification in accordance with the present invention then processes the received packets using the field or other parameter information in the packet. The field or other parameter information is matched to the filter-rule table entries to identify the filter-rule rectangle associated with the filter-rule to be applied to the packet.

In accordance with the present invention, values for each $s_m$ of $r_m=(s_m,d_m)$ in the s-dimension are desirably ranges that are a power of two. Consequently, prefix values ("prefixes") define ranges ("prefix ranges") that are a power of two. The length of a prefix is the number of specified bits of the prefix. The prefix range is between a lower bound defined by the prefix and unspecified bits set to logic "0" and the upper bound defined by the prefix and unspecified bits set to logic "1". The length may be represented by a binary value. The $d_m$ may be single points, ranges defined in a manner similar to prefix ranges in the s-dimension, and/or ranges defined as continuous ranges. When multiple matches of a same length prefix occur for a specific value of $s_m$, the query point (S, D) is associated with the highest priority filter rule having the matching prefix of $d_m$, if an overlap also occurs in the d-dimension.

FIG. 3 illustratively depicts prefixes and prefix ranges of a field in a s-dimension where the prefix ranges are a power of two. Field values s, which may be source addresses, vary from 000 to 111 (binary). An address may be a point (i.e., 010) or within a range (i.e., 010 to 101). For a special case, prefix ranges may be a power of 2. For example, if a prefix range is defined as Oxx, the prefix, represented as a single value 0, specifies the range 000to 011. For this example, the prefix has a length of 1 corresponding to one specified bit. Two prefixes of length 1 are possible: $I_1^0$ and $I_1^1$. If the prefix has two bits, or a length of 2, then four prefixes are possible: $I_2^0$, $I_2^1$, $I_2^2$, and $I_2^3$. Prefixes of different length define prefix ranges that are different powers of two. The prefix ranges do not overlap.

FIG. 4 illustrates an example of decomposition in the d-dimension of a 2-dimensional filter-rule rectangle into 1-dimensional overlapping segment sets and then into non-overlapping intervals. As described previously, values for each $d_m$ of filter rule $r_m=(s_m,d_m)$ in the d-dimension may be any contiguous range and are not necessarily restricted to prefix ranges only. FIG. 4 shows a horizontal axis 429 for the d-dimension representing, for example, parameter values for IP destination addresses. The process searches through each of the applicable filter rules $r_1, \ldots r_4$ to be implemented in the router for each dimension, and the process may be implemented before processing of arriving packets. Each of the filter rules $r_1, \ldots r_4$ specifies field ranges such as $d_1, \ldots d_4$ for the d-dimension applicable to the particular parameter of the packet header.

Field ranges $d_1, \ldots d_4$ are projected as overlapping horizontal line segments, with each segment specifying a start point "$b_i$" and end point "$q_i$" of a range for a particular corresponding filter rule (i an integer greater than 0). For example, $d_1$ specifies a first range of source addresses on a first segment defined by start point "$b_1$" and end point "$q_1$" for filter rule $r_1$. Segments may overlap, such as those of $d_1$ and $d_2$. Consequently, segments are decomposed into non-overlapping intervals $I_j$ (j an integer greater than 0). Therefore, the segment defined by start point "$b_1$" and end point "$q_1$" for filter rule $r_1$ has a single associated interval $I_1$, but the segment defined by start point "$b_2$" and end point "$q_2$" for filter rule $r_2$ has three intervals $I_1$, $I_2$, and $I_3$ associated with filter rule $r_2$. These three non-overlapping intervals $I_1$, $I_2$, and $I_3$ are a result of decomposing the overlapped segments of filter rules $r_1$, $r_2$, and $r_3$ at start or end points. It should be understood that for each filter rule, a range of source addresses and a range of destination addresses, for example, may be specified.

As described previously, values in the s-dimension of each rectangle desirably have lengths of a power of 2 when the values in the s-dimension are defined as prefix ranges. Ranges in dimensions being prefix ranges provide constraints such as illustrated in FIG. 3. When prefix range intervals have lengths which are powers of two, arbitrary overlapping of filter-rules for the dimension does not occur since two prefixes of the same length do not overlap. Also, a prefix range interval starts from an even-value point and terminates at an odd-value point. Consequently, a set of prefix ranges form several distinct cells distinguished by the length of the prefix or, equivalently, the length of the range. Further, values for each $d_m$ of filter rule $r_m=(s_m,d_m)$ in the d-dimension may be any contiguous range, such as illustrated in FIG. 4, and are not necessarily restricted to prefix ranges unless the value for $d_m$ is defined as a prefix range. However, modifying the packet filter in accordance with the present invention to define values for $d_m$ as prefix ranges may be desirable, such as if destination addresses are concatenated with layer-4 destination ports or some other similar header field.

In accordance with the present invention, filter-rule table cells for prefix ranges and associated non-overlapping intervals are defined containing pointers to filter-rules as entries in the filter-rule table in the following manner. Given each rule $r_i=(s_i,d_i)$, for the field range $s_i$ that is an integer power of 2, the length is defined as $l_{si}$ bits and for the field range $d_i$ the length is defined as $l_{di}$ bits. The maximum values of lengths $l_{si}$ and $l_{di}$ are defined as $l_{sMAX}$ and $l_{dMAX}$, respectively. The set of prefixes having a length of i bits are denoted as $P_i$, $i<0 \leqq l_{sMAX}$. As described with respect to FIG. 3, there may be several different prefixes of a given length i, i.e. the set of prefixes of length 1 ($P_1$) may have up to two elements, prefixes starting with "0" and prefixes starting with "1". The value $np_i$ denotes the number of elements in the set of prefixes of length i ($P_i$) that are present in the lookup table. The elements of the set of prefixes of length i ($P_i$) may be numbered in ascending order of their values; consequently, the $np_i$ prefixes of the set $P_i$ are defined as the set $\{P_i^1, P_i^2, \ldots, P_i^{npi}\}$.

The set of filter-rule rectangles $RP=\{RP_1, RP_2, \ldots, RP_{1sMAX}\}$ is defined such that each $RP_i$ is a subset of the set of n filter rule rectangles RP such that subset $RP_i$ includes all filter-rule rectangles formed from s value prefixes having a length of i bits. Further, each subset $RP_i$ may be defined as the union of the sets of filter-rule rectangles $RP_i^j=\{(P_i^j,d_i^1), (P_i^j,d_i^2), \ldots,\}$ where each filter-rule rectangle $RP_i^j$ has the $^{jth}$ prefix of length i ($P_i^j$) as a side of the filter-rule rectangle in the s-dimension. Therefore, each of the filter-rule rectangles in set $RP_i^j$ may associated with each prefix $P_i^j$ (j an integer and $1 \leqq j \leqq np_i$).

Each value $d_i^j$ in the d-dimension of the set of filter-rule rectangles $RP_i^j=\{(P_i,d_i^1), (P_i^j,d_i^2), \ldots, \}$ is a range in the d-dimension that may overlap other ranges. As defined, the subset of rectangles $RP_i$ is the union of sets $RP_i^1, RP_i^2, \ldots, RP_i^j \ldots RP_i^{npi}$, (j an integer and $1 \leqq j \leqq np_i$), and each of the $RP_i^j$ are disjoint. Filter-rule rectangles in set $RP_i^j$ are formed with longer prefixes than those filter rectangles in set $RP_i^j$ if i>t. A filter-rule having a longer prefix value in the s-dimension may be defined to have higher priority than other filter-rules with shorter prefix length since they are more specific with respect to, for example, packet source address. Consequently, if filter-rule rectangles in $RP_i^j$ and $RP_i^j$ match a point EP=(S, D) based on field values in the s-dimension, then the filter-rule associated with $RP_i^j$ is applied to packet EP. The filter-rule associated with $RP_i^j$ is applied to packet EP since rectangles in $RP_i^j$ are formed with longer prefixes than those rectangles formed in $RP_i^j$.

For the d-dimension, the size of the list of the set of $d_i^j$ values may be defined as $k_i^j$, k an integer greater than 1. From each list of j ranges in a rule set $RP_i$ comprising ($s_i$, $d_i^j$), a list of non-overlapping intervals $ID_i^j$ is formed along the axis of the d-dimension from filter-rule segments $Id_i^j$ corresponding to the values of $d_i^j$. The size of this new set of intervals $ID_i^j$ may be $K_i^j \leqq 2k_i^j+1$. By representing the original $k_i^j$ overlapping intervals as non-overlapping intervals, a memory space requirement of the packet filter may be increased by only a constant factor of 2.

For the d-dimension, if the values for $d_i^j$ are defined to be prefix ranges, then the projected filter-rule segments $Id_i^j$ along the d-dimension axis do not overlap, and so the $Id_i^j$ become the list of non-overlapping intervals $ID_i^j$.

For the general case, replacing overlapping intervals by non-overlapping intervals allows a search algorithm to locate the field value D from the query point (S, D) on one of these non-overlapping rectangles during the search procedure. The search algorithm then retrieves the associated enclosing rectangle of the non-overlapping rectangles representing the filter rule to be applied to the packet. Consequently, when many filter-rule rectangles overlap a given interval in the d-dimension, the particular filter-rule rectangle associated with the given interval when non-overlapping intervals are formed is the filter-rule rectangle with the highest priority that overlaps the interval.

FIG. 5 illustrates a 2-dimensional space for an exemplary packet filter in accordance with the first embodiment. FIG. 5 shows a total of $np_1=2$ prefixes of length i equal to 1 (i.e. 0xxx and 1xxx). For the set of rectangles $RP_1$ with prefix length i equal to 1, the corresponding set of filter-rule rectangles is $RP_1=\{e1,.e6\}$. Also shown is a total of $np_2=1$ prefixes of length i equal to 2 (i.e., 01xx) for the set $RP_2$ of filter-rule rectangles formed with prefixes of length i equal to 2. The set $RP_2$ includes the filter-rule rectangles {e2, e3, e4}. These filter-rule rectangles may overlap on the axis of the d-dimension. Similarly, set of filter-rule rectangles $RP_3$ with prefix of length i equal to 3 (i.e., 011x) contains one filter-rule rectangle e5.

For the illustration shown in FIG. 5, the set of intervals given a prefix length of 2 that are created after this overlap elimination for each $Id_2^1$ is $ID_2^1=\{a_0, a_1, \ldots a_6\}$. Filter-rule rectangles e2 and e3 overlap in the d-dimension. Filter-rule rectangle e3 of the set of rectangles $RP_2^1$ is associated with interval $a_2$, since this filter-rule rectangle may be defined to have the higher priority than filter rule rectangle e2.

Consequently, only this filter-rule rectangle e3 is associated with interval $a_2$ even though another filter-rule rectangle with lower priority overlaps this range $a_2$.

For the exemplary system of FIG. 5, a packet EP with header field values (S=0110, D=0101) arrives. First, a matching prefix of length 1 from S=(0) is found and a search performed for enclosing rectangles formed with this prefix. The d-dimension is searched and filter-rule rectangle e1 shown in FIG. 4 is a first candidate rule, or is the current solution. Note that rectangles e1 and e6 of FIG. 5 are the only rectangles in the set of rectangles with prefixes of length equal to 1. Next, a search for the matching prefix (01) is performed over the prefixes of length 2. Rectangle e3 is determined to be a better candidate rule since 1) the D value of the arriving packet overlaps with the range a2, 2) this filter-rule rectangle e3 is formed with a longer prefix than rule e1, and 3) this filter-rule rectangle has higher priority than other rectangles formed with prefixes of equal or lower length. Finally, a matching prefix (001) of length 3 is located and a search among rectangles with this prefix is performed, resulting in the rule of rectangle e5 as the best solution.

A packet filter of the present invention for a router employs an algorithm having two parts. The first-part-is-a-pre-processing-algorithm-that-searches-through-filter-rules and decomposes the filter rules for each dimension. The first part is performed by the router prior to processing of received packets. A second part is a classification algorithm that processes the received packets using the field or other parameter information in accordance with the processed filter rules of the pre-processing algorithm.

An exemplary pre-processing algorithm for a packet filter in accordance with the present invention is shown and described with respect to FIG. 6 and FIG. 7. The pre-processing algorithm performs three operations to decompose the n filter-rule rectangles. First, the filter-rule rectangles are separated based on the prefix length in the s-dimension. Second, for each prefix of length i, all associated filter-rule rectangles are projected onto the corresponding axis in the d-dimension to obtain first the overlapping intervals $Id_i^j$. Third, a set of non-overlapping intervals $ID_i^j$ are created from these the overlapping intervals $Id_i^j$. The non-overlapping intervals may be created by a scan of the overlapping intervals from lower to higher coordinates in the d dimension.

FIG. 6 illustrates a flowchart of an exemplary pre-processing algorithm in accordance with the present invention. First, at step **601** the set of prefixes $P_i^j$ (as defined previously) for all i and j, $1 \leq i \leq l_{sMAX}$ and $1 \leq j1 \leq np_i$, is stored in memory according to, for example, an efficient trie representation. Then, at step **602** for each filter-rule having prefix $P_i^j$, the corresponding set of filter-rule values $d_i^j$ in the d-dimension are projected as overlapping segments $Id_i^j$. At step **603**, for all $P_i^j$, (i.e., for all j prefixes of length i, $1 \leq i \leq l_{sMAX}$ and $1 \leq j1 \leq np_i$), the overlapping segments $Id_i^j$ are decomposed into a set of non-overlapping intervals $ID_i^j$. At step **604** a pointer is constructed to identify the highest priority filter-rule rectangle overlapping the associated non-overlapping interval for all intervals of the set $ID_i^j$. At step **605**, the set of non-overlapping intervals $ID_i^j$ are stored with associated prefix $P_i^j$ as table entry in the filter-rule table. Each entry of the filter-rule table corresponds to the pointer identifying actions to applied to a packet for a corresponding filter rule. The list of non-overlapping intervals $ID_i^j$ may be stored in sorted sequence using either an array or a binary tree. At step **606**, the algorithm returns to step **602** if $i<l_{sMAX}$, or until all prefix lengths $P_i$ are processed.

FIG. 7 is a flowchart illustrating the decomposition of intervals of the steps **603** and **604** of FIG. 6. For step **603** of FIG. 6, first, at step **701** the overlapping intervals $Id_i^j$ are sorted into an ascending sequence based on interval starting points. Then, at step **702**, for all j, if an overlapping interval $Id_i^j$ starts or ends, an assigned, non-overlapping interval $ID_i^j$ is generated for previous interval. For step **604** of FIG. 6, at step **703**, the assigned, non-overlapping intervals $ID_i^j$ and corresponding pointer to actions for the highest-priority filter-rule rectangle overlapping this interval are stored in memory. Optionally, at step **704** the newly created interval and the previously stored adjacent interval are compared, and are merged if the two intervals point to the same filter-rule. Since a new interval $ID_i^j$ is created, at most, when an overlapping interval begins or terminates, the size of this new set of intervals $ID_i^j$ is $K_i^j \leq 2k_i^j+1$ where $k_i^j$ is the size of the set of overlapping intervals $Id_i^j$.

In accordance with the pre-processing algorithm of the packet filter, each filter-rule is associated with a pointer in one or more filter-rule table entries. Each filter-rule pointer is stored in exactly one address in memory corresponding to prefix and prefix length on the s-dimension axis, and one or more addresses corresponding to non-overlapping intervals on the d-dimension axis. The set of filter-rule rectangles associated with a prefix is stored as a list of non-overlapping intervals and requires space only proportional to the size of the set. Only O(n) memory space may be utilized to store all the rectangles since each rectangle appears only in one set and therefore the size of the union of all sets is O(n).

Once the preprocessing algorithm creates the filter-rule table, the classification algorithm performs a look-up search of the filter-rule table. FIG. 8 illustrates an exemplary flow-chart of the classification algorithm of the packet filter. The classification algorithm may begin at step **801**. First, at step **801**, prefixes of length i, $P_i=\{P_i^1, P_i^2, \ldots, P_i^{np_i}\}$ are identified. Initially, the value of i may start from the shortest prefix length, such as i=1. Next, at step **802** the prefix $P_i^j$ of length i with an $s_i$ matching the query point S in the s-dimension is determined. If no match of S with $s_i$ in $P_i^j$ is found at step **802**, then the algorithm moves to step **805**. At step **805**, the prefix length value i is incremented, until the longest prefix length is searched (i.e. increment i if $i<l_{sMAX}$). Consequently, the classification algorithm repeats for each prefix length until all prefix lengths have been searched.

If a match of S with an $s_i$ in $P_i^j$ is found at step **802**, then at step **803** the stored structure in the d-dimension associated with $P_i^j$ is searched to find the non-overlapping interval $ID_i^m$ m that contains the query point D in the d-dimension. At step **804** the current solution is set as the pointer associated with table entry $(P_i^j, ID_i^m)$ (m an integer greater than 0). The current solution may be the "best" solution among all prefix lengths searched so far if shorter prefix lengths correspond to lower priority rules, and the search begins at the shortest prefix (lowest priority) and goes to the longest prefix (highest priority). The algorithm then moves to step **805**.

The number of iterations of the classification algorithm in the worst case is equal to the largest number of possible prefix lengths, which is $l_{sMAX}$. Consequently, the total time for searching through all prefix lengths is $O(l_{sMAX})$ times the time to search a list for a prefix length. In addition, the size of the lists of $ID_i^j$ for a prefix length may be O(n) since there are n filter-rules. Hence, an average O(log n) time is needed to search each list for a matching entry. The worst case total execution time of the exemplary classification algorithm is, therefore, $O(l_{sMAX} \log n)$.

However, for large numbers of table entries, worst case performance may not be sufficient for available processor speed. For example, if a number of possible prefix lengths

$l_{sMAX}$ is 32 and the number of table entries n is $2^{18}$=256K. This exemplary classification algorithm may perform 576 memory accesses in the worst case, which may be prohibitively high. An alternative embodiment of the present invention employs a trie structure with virtual intervals for storage of data in memory to reduce the worst-case time-complexity $O(l_{sMAX}\log n)$ to a time-complexity $O(l_{sMAX})$.

A trie structure may be employed for data storage with a memory space requirement that may be O(n). Furthermore, the order of search for the sets of filter-rules $RP_1$, $RP_2$, . . . , may be organized by increasing order of prefix lengths. For example, a set of intervals from $RP_1$ is searched before searching a set of intervals from $RP_2$ and so on. The search proceeds in levels $L_i$, with a search of sets belonging to $RP_1$ being on the first level $L_i$, those in $RP_2$ being on the second level $L_2$and so on. The number of non-overlapping intervals in all of $RP_i$ is defined as $N_i$. The root (i.e, bottom-most) level $R_{ie}$ has $N_{ie}$ non-overlapping intervals, and this level may be $RP_1$ with $N_1$ non-overlapping intervals. The number of overlapping intervals at each level without introducing virtual intervals may be O(n). In accordance with the present invention, introducing "virtual" intervals decreases search time of the classification algorithm in multiple ordered lists. If elements of a set of intervals are arranged by employing virtual intervals as described below, the worst case execution time may be $O(l_{sMAX}+\log n)$.

A search of the list of non-overlapping intervals at level $L_i$, for example, yields a result of the point D, where D is in an interval $ID_i^j$. A search of the lists at the next level $L_{i+1}$ is performed, instead of searching through the remaining intervals at level $L_i$. In general, the result of the previous search at level $L_i$ may be used for the search at level $L_{i+1}$, and the search at level $L_{i+1}$ is performed for only those intervals that fall in the range of intervals $ID_{i+1}^{j'}$ in level $L_{i+1}$ given by the interval $ID_i^j$ at $L_i$. For this case, since each level at level $L_{i+1}$ there may be O(n/ls) intervals which fall within the range determined by $ID_i^j$. Hence, an O(log(n/ls))=O(log n) search may be needed at every level.

Consequently, virtual intervals at levels $L_i \leq L_{lsMAX}$ are defined in the following manner. The number of intervals $N_i$ is defined at level $L_i$. Boundary points that demarcate the $N_i$ intervals in the d dimension at level $L_i$ are denoted by $y_1^i$, $y_2^i$ . . . with a maximum of $2N_i$ such points. Every other point at level $L_i$ is replicated at level $L_{i-1}$, and up to $2N_i$ points are so propagated to level $L_{i-1}$. Although the present embodiment is described using propagation of every other point, other embodiments may skip NS points, NS an integer greater than 1, or may vary the number of points skipped according to granularity of the pointers used.

The points that were propagated together with the points defining original non-overlapping intervals $ID_i^j$, now define intervals at level $L_{i-1}$ as new intervals $VD_{i-1}^{j'}$. These intervals are stored as non-overlapping intervals at level $L_{i-1}$. Next, for all the intervals at level $L_{i-1}$ and their associated points, every other point is replicated and propagated as virtual points to level $L_{i-2}$. This propagation process is repeated until the root level $L_{ie}$, (i.e., $L_1$) is reached. Note that the propagation process is employed to speed up the search; at each level, the filter-rule rectangles associated with each non-overlapping interval are as described in the preprocessing algorithm described previously. Virtual intervals and points that result from propagation are desirably ignored for association of filter-rule rectangles with non-overlapping intervals.

The propagation process increases memory space requirements by a constant factor, and so the total memory space requirement is still O(n). A maximum amount of virtual intervals created and corresponding maximum memory space is when $N_{lsMAX}$=n, n being the number of filter rules, in which case the number of boundary points at level $L_{sMAX}$ is 2n. The extra memory space due to the propagations is then as given in equation (1)

$$\left(n+\frac{n}{2}+\frac{n}{4}+\ldots\right)\leq 2n \tag{1}$$

Increasing the memory space by a constant factor, however, allows for searching of multiple lists (i.e. lists of non-overlapping intervals at each level) efficiently. A packet EP=(S, D) arrives at the packet filter and is processed by the classification algorithm with a filter-rule table organized in accordance with the alternative embodiment. A first level, i.e., $L_1$ list of non-overlapping intervals $VD_i^j$ is searched as described previously with respect to the classification algorithm, taking O(log n) time for the worst case. This search results in locating the given point D in an interval $VD_i^j$ that may be a virtual interval propagated from the level $L_2$. With D localized to this interval $ID_i^j$, a search in the next level $L_2$searches in the range of intervals given by $VD_1^j$. Because every other point has been propagated up from level $L_2$, only 2 intervals in $VD_2^j$ may fall within the interval $VD_1^j$ to which D has been localized. Hence, the search at level $L_2$ may be completed in O(1) time. In general, in moving from level $L_i$ to level $L_{i+1}$, the propagation of intervals allows enough information gained in the search at level $L_i$ to be employed in the search at level $L_{i+1}$ is O(1) time. Hence, the worst case execution time of the look-up algorithm of the alternative embodiment is $O(l_{sMAX}+\log n)$.

FIG. 9A and 9B illustrate an example of an alternative embodiment of the packet filter employing virtual intervals to reduce search time of the classification algorithm. FIG. 9A illustrates a trie structure employed to search prefix values of fourteen exemplary filter rules in ascending order of length. FIG. 9B shows creation of virtual intervals for levels of a portion of the trie structure shown in FIG. 9B. For the exemplary embodiment of FIG. 9A and FIG. 9B, Table 1 provides a list of filter-rules with corresponding prefix values and lengths for source fields and destination field ranges.

TABLE 1

| Filter-Rule Number | Source Prefix Value | Prefix length | Destination range d (lower bound, upper bound) |
|---|---|---|---|
| | 11* | 2 | (0,15) |
| 2 | 0* | 1 | (4,7) |
| 3 | 00* | 2 | (12,15) |
| 4 | 0* | 1 | (12,15) |
| 5 | 0* | 1 | (8,15) |
| 6 | 10* | 2 | (8,15) |
| 7 | 001* | 3 | (8,15) |
| 8 | 000* | 3 | (6,7) |
| 9 | 000* | 3 | (4,5) |
| 10 | 001* | 3 | (8,9) |
| 11 | 001* | 3 | (4,5) |
| 12 | 001* | 3 | (10,11) |
| 13 | 001* | 3 | (12,13) |
| 14 | 001* | 3 | (0,3) |

A packet EP with fields S=0010 and D=1101 arrives in the system. Referring to FIG. 9A, a search of the trie structure 900 (the trie search) in the s-dimension begins at the root level 901 (level 0) to determine if the source address (S=0xxx) begins with a 0(state 902) or a 1 (state 903). This is a search of the set of prefixes of length 1. The trie search

moves to the state **902** at level 1 corresponding to the prefix 0xxx of length 1. Similarly, at level 2 the trie search determines if the next bit of the source address (S=00xx) is a 0 (state **904**) or a 1 (state **905**). The trie search moves to the state **904** at level 2 corresponding to the prefix 00xx of length 2. Finally, at level 3 the trie search of a portion of the set of prefixes of length 3 determines if the next bit of the source address (S=001x) is a 0 (state **908**) or a 1 (state **909**). The trie search moves to the state **909** at level 3 corresponding to the prefix 001x of length 3. For searches of prefixes, only a portion of sets of prefixes are searched in the tries. Consequently, states **903**, **906** and **907** are not reached since the trie search moves from state **901** to state **902**, to state **904**.

FIG. 9B illustrates an example of virtual intervals and point propagation to reduce search time of the classification algorithm. First, non-overlapping intervals in the d-dimension are shown for selected states at each level. For example, at level 1, state **902** corresponds to the prefix of length 1 being 0xxx. The filter-rules of this prefix 0xxx (from Table 1) are rules 2, 4 and 5 with respective filter-rule segments (decimal ranges in the d-dimension) of (4,7), (8,12) and (8,15). These filter-rule segments are then decomposed into non-overlapping intervals (4,7), (8,12) and (12, 15). Without virtual intervals, the trie search at level 1 searches these three intervals to find the value D=1101 (i.e., 13 decimal) included in the third non-overlapping interval (12,15) associated with rule 5. However, for the next level 2, the information of this search is lost.

Referring to FIG. 9B, the non-overlapping intervals of the highest level, level 3, are shown for the states **908** and **909**. Points of these original, non-overlapping intervals at level 3 are propagated to the non-overlapping intervals at level 2. Brackets in FIG. 9B indicate original, non-overlapping intervals. For the example shown, alternate points of the intervals of the left state **908** (next bit **0**) and right state **909** (next bit 1) are inserted into the non-overlapping intervals of the states of the next level 2, but as described previously the present invention is not so limited. For example, virtual intervals (0,3), (3,4), (5,6), (6,9), (9,11), (11,12), (12,13) and (13,15) are created from the original non-overlapping interval (12,15). Next, the alternate points of the intervals of state **904** are propagated to level 1, and as shown, propagated points, such as 12, may be duplicated in a level, since pointers are to be associated with the intervals. Normally, points of left and right states are propagated, but for the example of FIG. 9A and FIG. 9B, no rules or intervals are associated with state **905**.

As the trie search of prefixes as shown in FIG. 9A progresses, the search of intervals is as shown in FIG. 9B. At level 1, state **902**, the intervals in the d-dimension are searched and the value of D=1101, 13 decimal, is determined to be included in the interval (12, 12,15). At level 2, after the prefix search moves to state **904**, the pointer associated with propagated point 12 in interval (12,12,15) is employed to limit the search in level 2 to interval (12,13,15). At level 3, after the prefix search moves to state **909**, the pointer associated with propagated point 13 in interval (12,13,15) is employed to limit the search in level 3 to interval (12,13), associated with rule 13 of Table 1.

As described, the algorithm for computing the filters is largely implemented in hardware and may be manufactured in application specific integrated circuit (ASIC) form, or as a field programmable gate array (FPGA) that consequently, may operate at very high speed. FIG. 10 illustrates the hardware system **1000** for implementation of the packet filter in accordance with the present invention in a packet forwarding engine or router, including an input line **1005** for receiving an incoming packet and a bi-directional CPU interface line **1010** representing control and timing lines for purposes of illustration. The incoming packet is input to a pipeline register **1025** for temporary storage and is also input to each classification processor **1050**. Classification processor **1050** employs memory **1030** to identify a filter-rule to be applied to the incoming packet. Field processor **1035** updates fields of the packet stored in pipeline register **1025** based on the identified filter-rule to be applied to the incoming packet. The details of classification processor **1050** are now described with reference to FIG. 11.

FIG. 11 shows a classification processor **1050** that receives the incoming packet and stores field parameters, e.g., source address and destination addresses S and D, in a register **1176**. Under the control of filter processor **1160**, optional memory control device **1165**, and associated memory **1030**, the search of the classification algorithm is performed whereby non-overlapping interval information from memory **1030** is provided to the register **1179** for each prefix length. Comparator **1180** performs a comparison to ascertain each interval associated with the D value of the received packet. After the correct solution for a filter-rule rectangle is found, its corresponding bitmap vector containing potential filter-rule actions is provided from register **1179** along line **1190**. From the resultant bitmap vector, the CPU will apply the rule of highest priority, and performs the action dictated by the filter rule upon the received packet stored in the pipeline register **1025**. Thus, the packet may be dropped or forwarded to another destination on output line **1015**.

The preprocessing algorithm of the present invention may be implemented in the classification processor by filter-rule processing and table processing modules. The filter-rule processing module may assign filter-rules to prefix values and lengths in one dimension, project the filter-rule segments in the other dimension, and decompose the filter-rule segments into non-overlapping intervals. The table-processing module may be employed to coordinate memory organization and storage, generating the necessary pointers with non-overlapping intervals for particular prefix value addressing schemes.

An example memory organization for the system is illustrated in FIG. 12, which depicts a filter-rule table having a plurality of interval lists in one dimension corresponding to each prefix length of another dimension, which may be associated with the following respective filter parameters: 1) destination addresses, and 2) source address. Entries of the filter-rule table are generated as described previously, i.e., with respect to FIGS. 6 and 7, and addressed by prefix values **1259***a*–**1259***d*. Each filter-rule table is shown to include an array **1260***a*–**1260***d* of intervals to be searched corresponding to prefix values as described above with reference to FIG. 8, and the corresponding filter actions **1261***a*–**1261***d* and the pointers **1262***a*–**1262** *d*.

While embodiments of the present invention are shown and described with respect to searches in a given dimension ordered from shortest to longest length, as would be apparent to one skilled in the art the present search algorithms and/or filter-rule table structures may be varied. For example, the search may be from the longest to the shortest prefix length, or from initial to final prefix values in an ordered list of the set of prefix values. Further, matching of packets field values with prefix values and interval values are described herein using binary search techniques, but the present invention is not so limited. As would be apparent to one skilled in the art, other search techniques to match values may be employed, such as employing a perfect hash method.

It will be further understood that various changes in the details, materials, and arrangements of the parts which have been described and illustrated in order to explain the nature of this invention may be made by those skilled in the art without departing from the principle and scope of the invention as expressed in the following claims.

What is claimed is:

1. Apparatus for associating at least one filter rule with a packet, each filter rule and the packet characterized by values in first and second dimensions, the filter rule to be applied to the packet by a router in a communications network, the apparatus comprising:

a storage medium adapted to store a filter-rule table, each entry of the filter-rule table corresponding to a prefix value having a length in the first dimension and at least one interval in the second dimension; and

a classification processor comprising:

a comparator adapted to identify each prefix value matching the value of the packet in the first dimension, and

a filter processor adapted to retrieve, from the filter-rule table, each interval associated with each prefix value identified by the comparator containing the value of the packet in the second dimension,

wherein the filter processor identifies as a solution interval the interval associated with the prefix length characterized by an associated predetermined metric and containing the second field, and

wherein the classification processor associates the filter rule corresponding to the solution interval with the packet.

2. The invention as recited in claim 1, wherein the classification processor further comprises a pre-processor including:

a filter-rule processing module adapted to:

assign each filter-rule to one or more prefix values based on the values in the first dimension,

project, for each prefix value having the same length, values of each corresponding filter rule of the prefix value onto the second dimension to define at least one filter-rule segment, and

decompose each filter-rule segment into one or more non-overlapping intervals associated with each prefix value of the same length in the second dimension; and

a table-processing module adapted to generate a pointer for each corresponding non-overlapping interval to identify an included filter-rule, the table-processing module adapted to store the pointer as an entry of the filter-rule table associated with a prefix value length and a non-overlapping interval.

3. The invention as recited in claim 2, wherein:

the filter-rule processing module further comprises:

assigning means for assigning each prefix value of the same length to a corresponding level;

first projecting means for projecting, for the level having prefix values of a first length, values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment;

second projecting means for projecting, in each level beginning at the level having prefix values having a second length, 1) values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment in a current level, and 2) selected points of the at least one non-overlapping interval in the previous level so as to define at least one virtual interval in the second dimension; and

interval forming means for forming each filter-rule segment and each virtual interval of the current level into one or more non-overlapping intervals associated with each prefix value having the same length.

4. The invention as recited in claim 3, wherein the first and second lengths are either 1) the longest and next longest lengths in a descending prefix length order, respectively, or 2) the shortest and next shortest lengths in an ascending prefix length order, respectively.

5. The invention as recited in claim 3, wherein the second projecting means projects, as selected points, every Nth point that defines either a start point or a stop point of each non-overlapping interval in the previous level, N an integer greater than 1.

6. The invention as recited in claim 2, wherein the values of each filter rule in the second dimension are at least one range being a power of 2, each range being projected as a corresponding filter-rule segment to form the non-overlapping interval in the second dimension.

7. The invention as recited in claim 1, wherein the values of each filter rule are field ranges, the field ranges in the first dimension being a power of two, and each prefix length defines a number of specified bits of the field range.

8. The invention as recited in claim 1, wherein an entry of the filter-rule table of the storage medium includes a pointer identifying at least one filter rule contained in the corresponding non-overlapping overlapping interval.

9. The invention as recited in claim 8, wherein each filter-rule has an associated priority, and the pointer identifies the filter-rule with the highest associated priority contained in the corresponding non-overlapping interval.

10. The invention as recited in claim 8, wherein the values of each filter rule are field ranges, the field ranges in the first dimension being a power of two, and each prefix length defines a number of specified bits of the field range.

11. The method as recited in claim 1, wherein the associated predetermined metric is either the prefix value having the longest prefix length, the shortest prefix length or the prefix length having a highest priority.

12. A method of associating at least one filter rule with a packet, each filter rule and the packet characterized by values in first and second dimensions, the filter rule to be applied to the packet by a router in a communications network, the method comprising the steps of:

a) providing a filter-rule table, each entry of the filter-rule table corresponding to a prefix value having a length in the first dimension and at least one interval in the second dimension;

b) identifying each prefix value matching the value of the packet in the first dimension;

c) retrieving, from the filter-rule table, each interval associated with each prefix value identified in step b) containing the value of the packet in the second dimension;

d) identifying, as a solution interval, the interval associated with the prefix value characterized by an associated predetermined metric and containing the value of the packet in the second dimension; and

e) associating the filter rule corresponding to the solution interval with the packet.

13. The method as recited in claim 12, wherein the step a) comprises the steps of:

f) assigning each filter-rule to one or more prefix values based on the values in the first dimension;

g) projecting, for each prefix value having the same length, values of each corresponding filter rule of the prefix value onto the second dimension to define at least one filter-rule segment;

h) decomposing each filter-rule segment into one or more non-overlapping intervals associated with each prefix value of the same length in the second dimension;

i) generating a pointer for each corresponding non-overlapping interval to identify an included filter-rule; and

j) storing the pointer as an entry of the filter-rule table associated with a prefix value length and a non-overlapping interval.

**14**. The method as recited in claim **13**, wherein:

step g) further comprises the steps of:

g1) assigning each prefix value of the same length to a corresponding level;

g2) projecting, for the level having prefix values having a first length, values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment,

g3) projecting, in each level beginning at the level having prefix values having a second length, 1) values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment in a current level, and 2) selected points of the at least one non-overlapping interval in the previous level so as to define at least one virtual interval in the second dimension; and

step h) further comprises the step of:

h1) forming each filter-rule segment and each virtual interval of the current level into one or more non-overlapping intervals associated with each prefix value having the same length.

**15**. The method as recited in claim **14**, wherein, for steps g2) and g3), the first and second lengths are either 1) the longest and next longest lengths in a descending prefix length order, respectively, or 2) the shortest and next shortest lengths in an ascending prefix length order, respectively.

**16**. The method as recited in claim **14**, wherein step g3) projects, as selected points, every Nth point that defines either a start point or a stop point of each corresponding non-overlapping interval in the previous level, N an integer greater than 1.

**17**. The method as recited in claim **13**, wherein the values of each filter rule in the second dimension are at least one range being a power of 2, the projecting step g) projects each range as a corresponding filter-rule segment in the second dimension, and the decomposing step h) forms the non-overlapping interval from the corresponding filter-rule segment projected in step g).

**18**. The method as recited in claim **12**, wherein the values of each filter rule are field ranges, the field ranges in the first dimension being a power of two, and each prefix length defines a number of specified bits of the field range.

**19**. The method as recited in claim **12**, wherein, for the filter-rule table provided in step a), an entry of the filter-rule table associated with a prefix value length and a non-overlapping interval includes a pointer identifying at least one filter rule contained in the corresponding non-overlapping interval.

**20**. The method as recited in claim **19**, wherein each filter-rule has an associated priority, and the pointer generated in step i) identifies the filter-rule with the highest associated priority contained in the corresponding non-overlapping interval.

**21**. The method as recited in claim **19**, wherein the values of each filter rule are field ranges, the field ranges in the first dimension being a power of two, and each prefix length defines a number of specified bits of the field range.

**22**. The method as recited in claim **12**, wherein for step d) the associated predetermined metric is either the prefix value having the longest prefix length, the shortest prefix length or the prefix length having a highest priority.

**23**. A method of storing at least one filter rule with values associated with first and second dimensions in a filter-rule table comprising the steps of:

a) assigning each filter-rule to one or more prefix lengths based on the values in the first dimension;

b) projecting, for each prefix length, values of each corresponding filter rule of the prefix length onto the second dimension to define at least one filter-rule segment,

c) decomposing each filter-rule segment into one or more non-overlapping intervals associated with each prefix length and corresponding filter rule in the second dimension;

d) generating a pointer for each corresponding non-overlapping interval to identify an included filter-rule; and

e) storing the pointer as an entry of the filter-rule table associated with a prefix length and a non-overlapping interval.

**24**. The method as recited in claim **23**, wherein:

step b) further comprises the steps of:

b1) assigning each prefix value of the same length to a corresponding level;

b2) projecting, for the level having prefix values of a first length, values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment,

b3) projecting, in each level beginning at the level having prefix values having a second length, i) values of each corresponding filter rule onto the second dimension to define at least one filter-rule segment in a current level, and ii) selected points of the at least one non-overlapping interval in the previous level so as to define at least one virtual interval in the second dimension; and

step c) further comprises the step of:

c1) forming each filter-rule segment and each virtual interval of the current level into one or more non-overlapping intervals associated with each prefix value having the same length.

**25**. The method as recited in claim **24**, wherein, for steps b2) and b3), the first and second lengths are either 1) the longest and next longest lengths in a descending prefix length order, respectively, or 2) the shortest and next shortest lengths in an ascending prefix length order, respectively.

**26**. The method as recited in claim **24**, wherein step b3) projects, as selected points, every Nth point that defines either a start point or a stop point of each corresponding non-overlapping interval in the previous level.

**27**. The method as recited in claim **23**, wherein the values of each filter rule are field ranges, the field ranges in the first dimension being a power of two, and each prefix length defines a number of specified bits of the field range.

**28**. The method as recited in claim **23**, wherein each pointer stored in the filter-rule table in step e) identifies each filter rule contained in the non-overlapping interval.

**29**. The method as recited in claim **23**, wherein each pointer stored in the filter-rule table in step e) identifies the filter-rule with the highest associated priority contained in the corresponding non-overlapping interval.

**30**. The method as recited in claim **23**, wherein the values of each filter rule in the second dimension are at least one range being a power of 2, the projecting step b) projects each range as a corresponding filter-rule segment in the second dimension, and the decomposing step c) forms the non-overlapping interval from the corresponding filter-rule segment projected in step b).

* * * * *

TOS

US006353616B1

(12) **United States Patent**
**Elwalid et al.**

(10) **Patent No.: US 6,353,616 B1**
(45) **Date of Patent: Mar. 5, 2002**

(54) **ADAPTIVE PROCESSOR SCHEDULOR AND METHOD FOR RESERVATION PROTOCOL MESSAGE PROCESSING**

(75) Inventors: **Anwar I. Elwalid**, Murray Hill; **T. V. Lakshman**, Eatontown, both of NJ (US); **Martin May**, Sophia Antipolis (FR)

(73) Assignee: **Lucent Technologies Inc.**, Murray Hill, NJ (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/221,552**

(22) Filed: **Dec. 28, 1998**

**Related U.S. Application Data**

(60) Provisional application No. 60/086,246, filed on May 21, 1998.

(51) **Int. Cl.**[7] ..... **H04B 7/212**

(52) **U.S. Cl.** ..... **370/443**; 370/389; 370/395.2; 370/412; 709/103; 709/105; 709/226

(58) **Field of Search** ..... 370/443, 522, 370/412, 468, 428, 429, 389, 395.2; 709/103, 104, 105, 226, 241

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | | |
|---|---|---|---|---|
| 5,774,668 A | * | 6/1998 | Choquier | 709/103 |
| 5,915,095 A | * | 6/1999 | Miskowiec | 709/103 |
| 6,167,049 A | * | 12/2000 | Pei | 370/395.2 |
| 6,262,976 B1 | * | 7/2001 | McNamara | 370/389 |
| 2001/0018701 A1 | * | 8/2001 | LiVecchi | 709/105 |

* cited by examiner

*Primary Examiner*—Brian A. Zimmerman
(74) *Attorney, Agent, or Firm*—Steve Mendelsohn; Ian M. Hughes

(57) **ABSTRACT**

A packet network employing a reservation-based protocol system includes routers having processing sections that schedule message processing of the protocol's control messages adaptively based on link utilization. A scheduler of the processing section employs a round-robin scheduling with adaptive weight assignment to allocate processing capacity for control messages. For the RSVP protocol, for example, messages are grouped in classes, and link utilization of the packet flows for each message class is monitored. Weights corresponding to a portion of the processing section's processing capacity are allocated to each message class. The weights are defined based on link utilization for the message class and average message queue length. For processing sections monitoring multiple links, weights are further defined for super-classes based on overall link utilization. Weights may change as link utilization and average message size changes. With defined weights adaptively defined, the processing section then processes each message class in a cyclic, "round-robin" fashion.

**15 Claims, 6 Drawing Sheets**

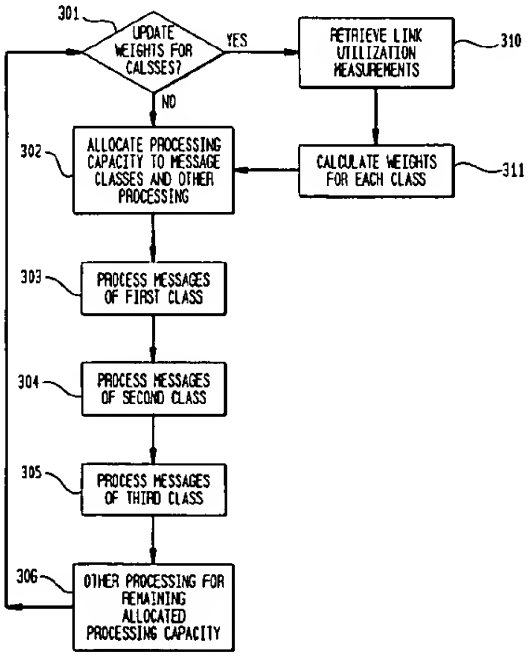


FIG. 1

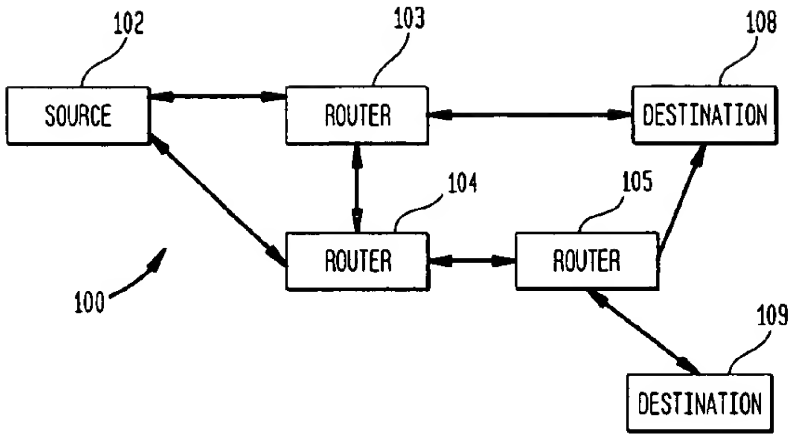

102
103
108
SOURCE
ROUTER
DESTINATION
104
105
ROUTER
ROUTER
100
109
DESTINATION

FIG. 2

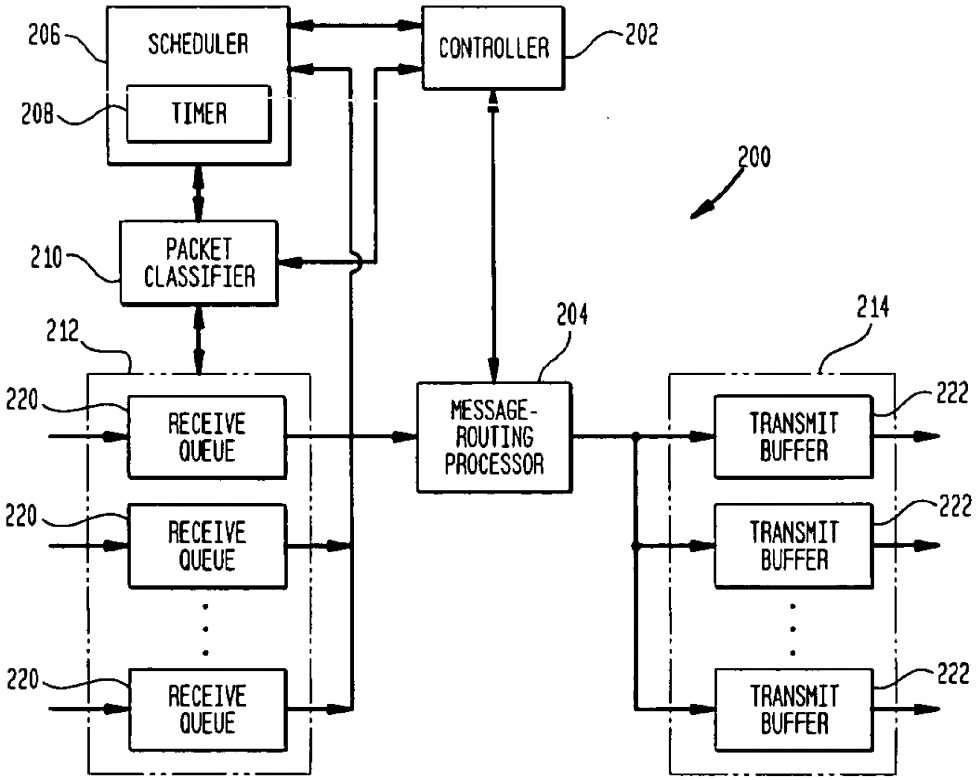

206
SCHEDULER
CONTROLLER
202
208
TIMER
200
210
PACKET CLASSIFIER
212
204
214
220
RECEIVE QUEUE
MESSAGE-ROUTING PROCESSOR
TRANSMIT BUFFER
222
220
RECEIVE QUEUE
TRANSMIT BUFFER
222
220
RECEIVE QUEUE
TRANSMIT BUFFER
222

FIG. 3

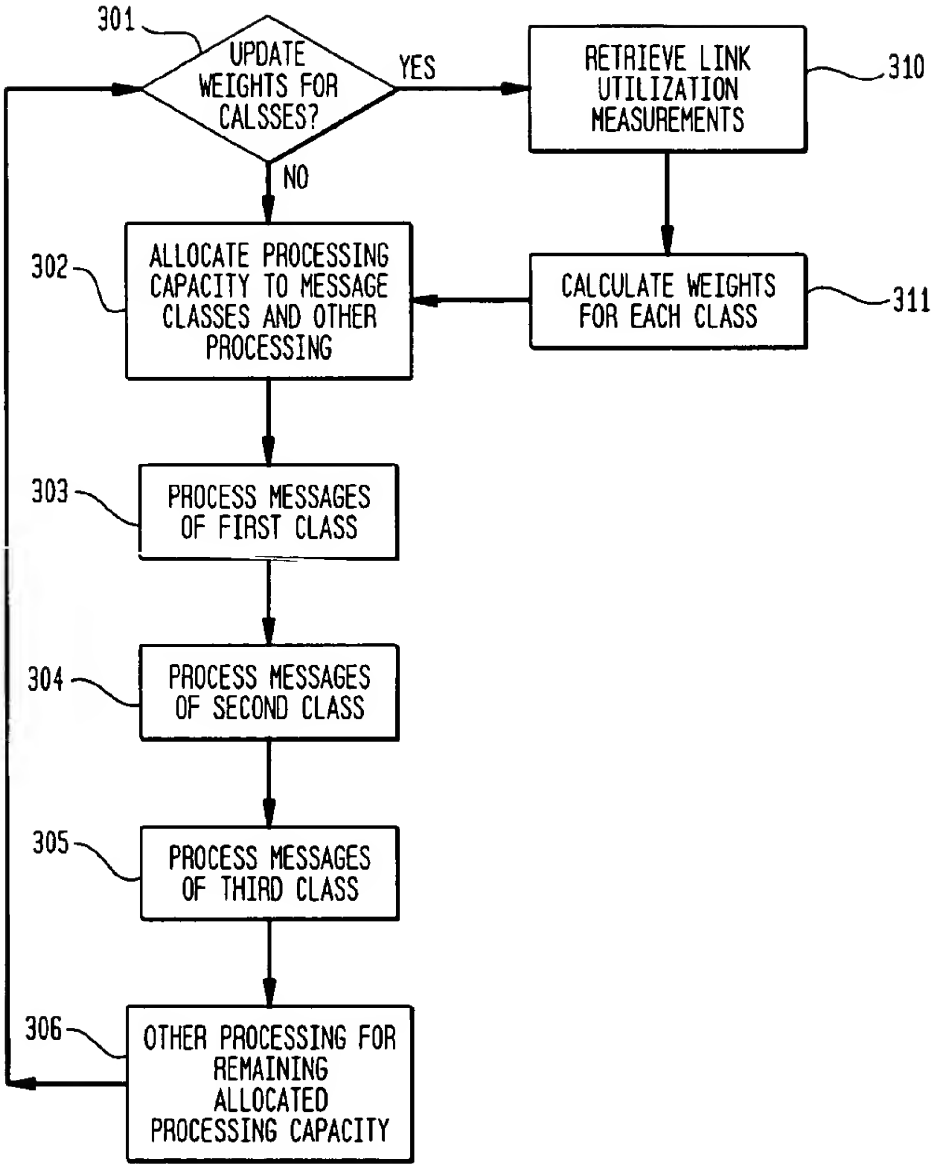

301
UPDATE WEIGHTS FOR CALSSES?
YES
NO
RETRIEVE LINK UTILIZATION MEASUREMENTS
310
CALCULATE WEIGHTS FOR EACH CLASS
311
302
ALLOCATE PROCESSING CAPACITY TO MESSAGE CLASSES AND OTHER PROCESSING
303
PROCESS MESSAGES OF FIRST CLASS
304
PROCESS MESSAGES OF SECOND CLASS
305
PROCESS MESSAGES OF THIRD CLASS
306
OTHER PROCESSING FOR REMAINING ALLOCATED PROCESSING CAPACITY

FIG. 4A
(PRIOR ART)

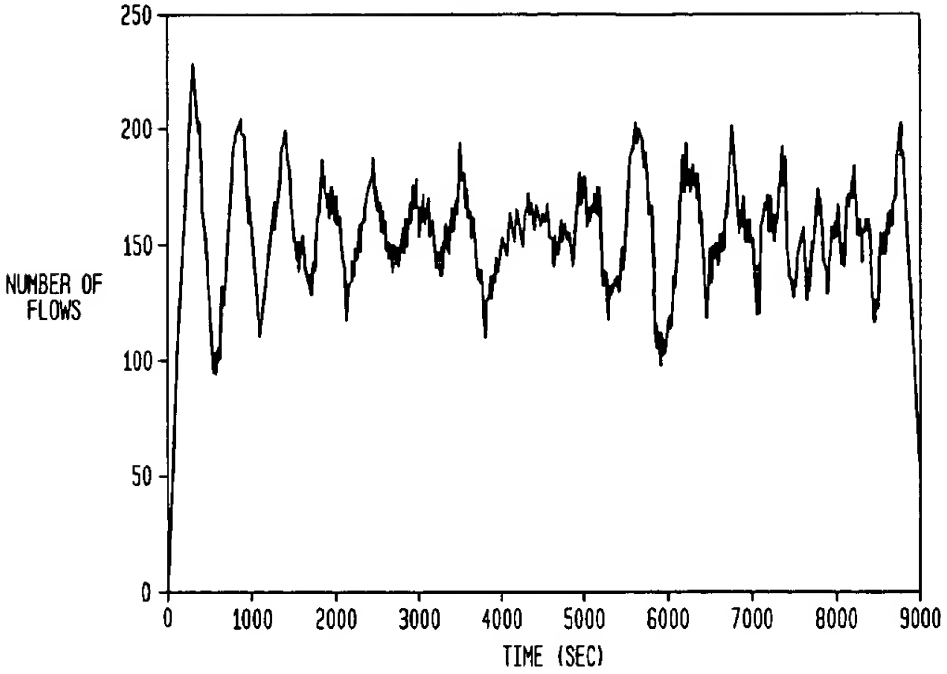



FIG. 4B
(PRIOR ART)

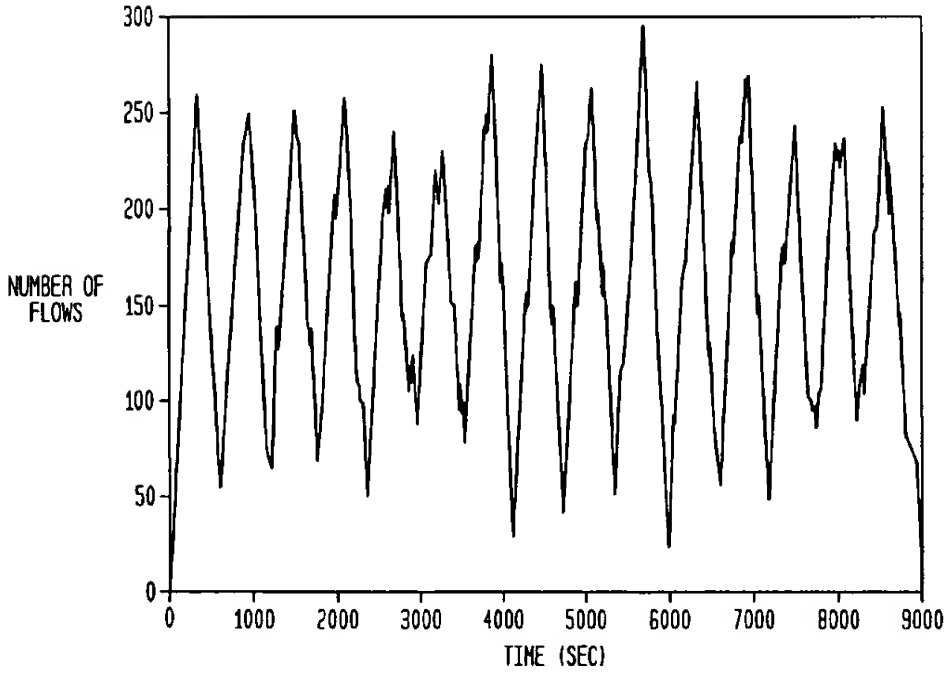


FIG. 5A
(PRIOR ART)

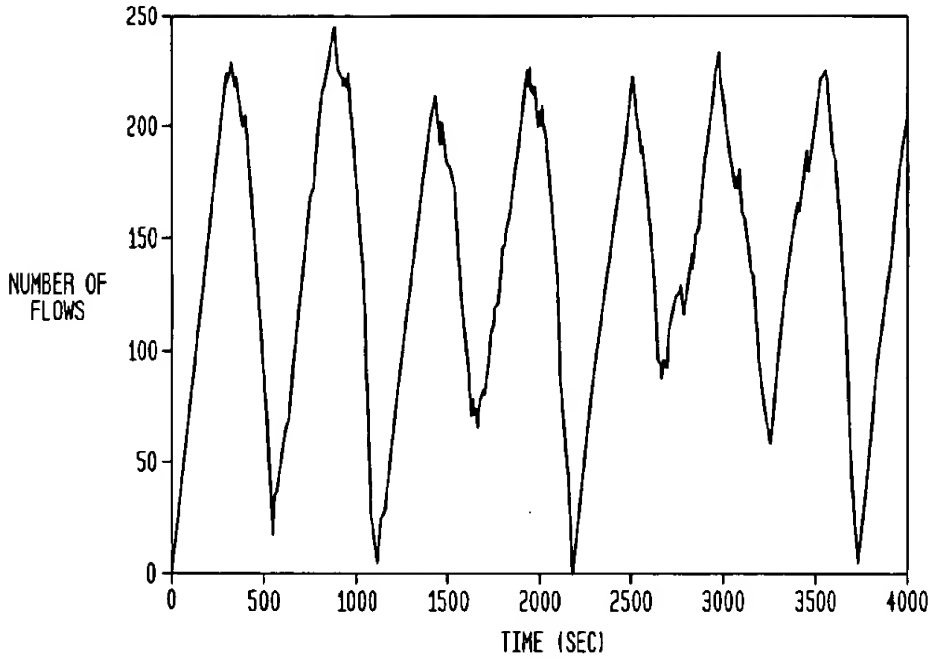



FIG. 5B
(PRIOR ART)

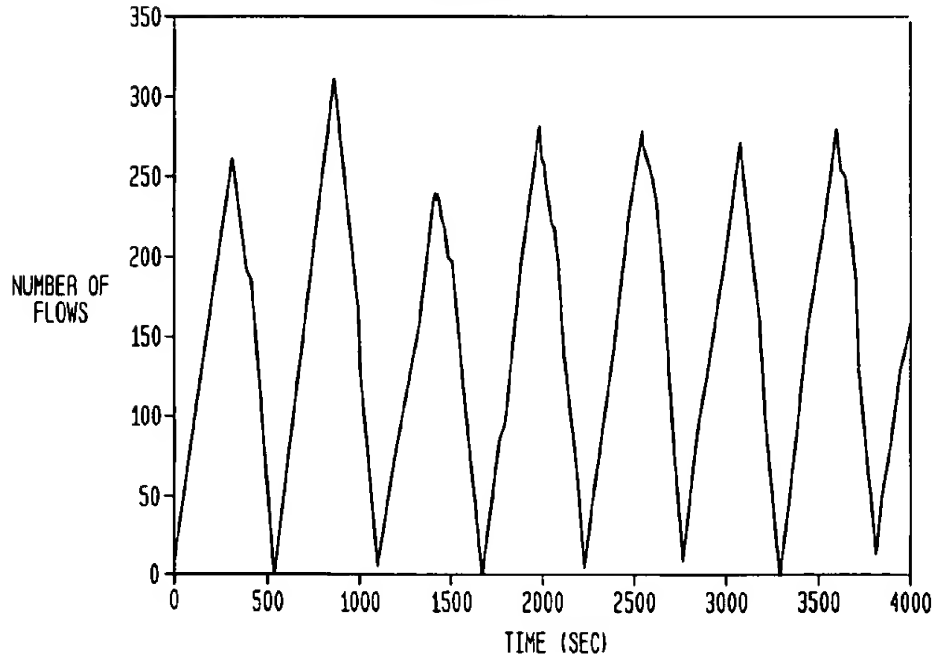


FIG. 6

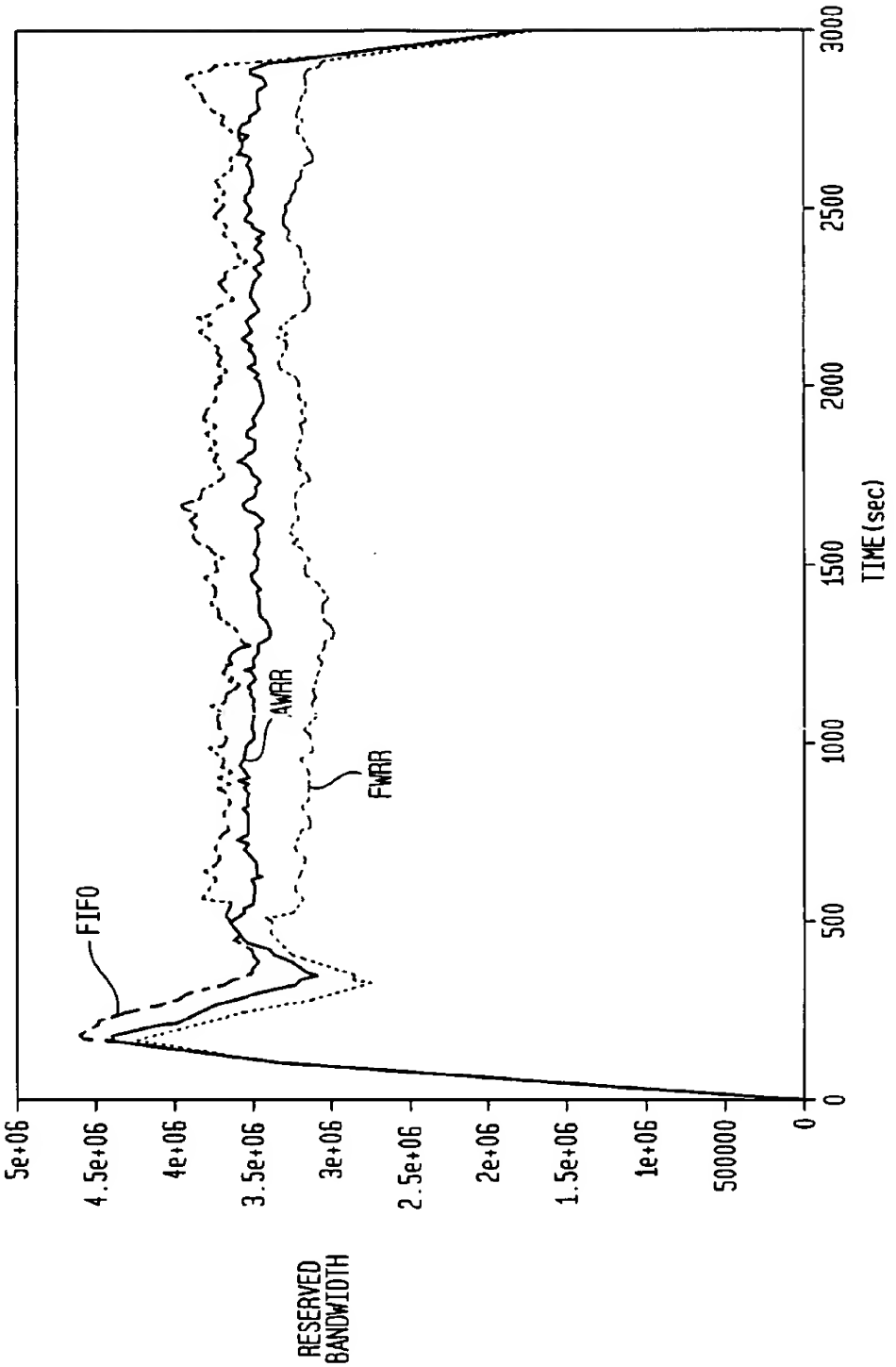
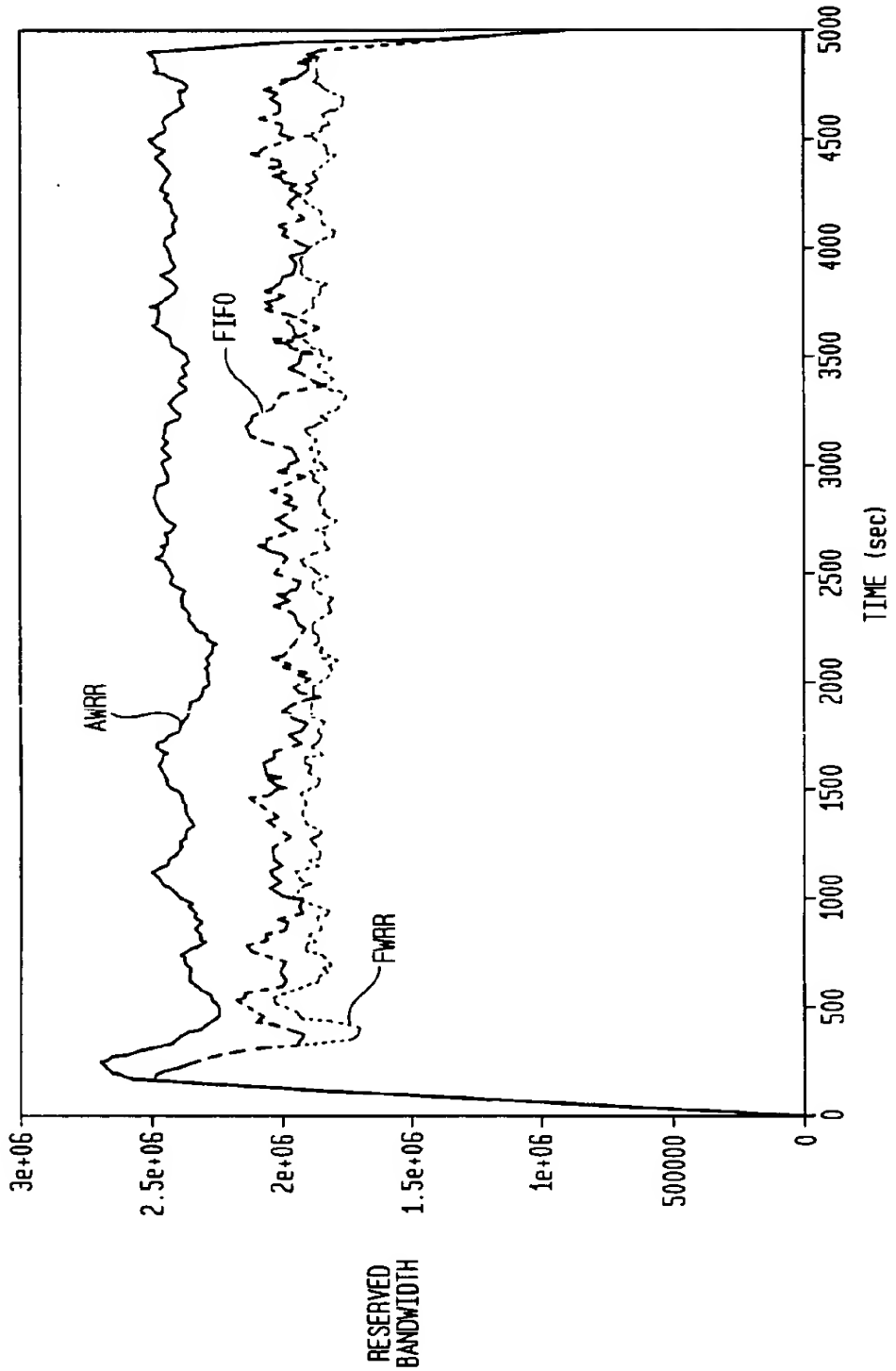

FIFO
AWRR
FWRR
RESERVED BANDWIDTH
5e+06
4.5e+06
4e+06
3.5e+06
3e+06
2.5e+06
2e+06
1.5e+06
1e+06
500000
0
0
500
1000
1500
2000
2500
3000
TIME(sec)

FIG. 7


AWRR
FIFO
FWRR
RESERVED BANDWIDTH
3e+06
2.5e+06
2e+06
1.5e+06
1e+06
500000
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
TIME (sec)

# ADAPTIVE PROCESSOR SCHEDULOR AND METHOD FOR RESERVATION PROTOCOL MESSAGE PROCESSING

## CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of the filing date of U.S. provisional application No. 60/086,246, filed on May 21, 1998.

## BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to packet networks, and, more particularly, to scheduling of control protocol message processing by a router.

2. Description of the Related Art

Packet networks, such Internet Protocol (IP) based networks, are increasingly providing differentiated services. One approach for providing differentiated services employs type-of-service (TOS) bits defined in the packet header. Routers within the packet network interpret the TOS bits in a predetermined manner so as to provide the differentiated services. Another approach, which is a reservation-based approach, employs control messages to reserve network resources for the duration of a connection defined by a packet flow, or packet flow aggregates ("flows"). For this reservation-based approach, a protocol that may be employed to signal reservation of network resources is the Reservation Setup Protocol (RSVP). RSVP, as an example, may be used in conjunction with service models, such as guaranteed rate and controlled load service models, to request a desired quality of service (QOS) for certain packet flows.

RSVP is a receiver-oriented resource reservation protocol: reservations are initiated when a source (sender) requests a resource reservation, such as a reservation for a certain amount of bandwidth of a transmission line or logical link during connection set-up or during an established connection. This RSVP request is signaled through the network using a PATH message. The PATH message is routed along the network to its destination (or set of destinations) through a series of routers in a similar manner to that of other IP packets. However, before propagating a PATH message, each router checks if sufficient requested resources are available. If the requested resources are available, the router first establishes a flow-state for the packet flow (or aggregated flows) indicated by this request and then propagates the PATH message. The packet flow in progress is maintained by periodic UPDATE messages generated by the source. Also, each intermediate router, and the destination, starts a counter, or refresh timer, that is employed to generate a processor interrupt causing termination of the packet flow if no RESV or UPDATE messages are received for the packet flow before timer expires.

When the PATH message reaches the desired destination (recipient), the destination sends back a RESV message through the network to the source. The RESV message may be, for example, a bandwidth request that may be different from the bandwidth requested in the PATH message. When an intermediate router of the network receives a RESV message for which there is an established packet flow for a connection, the intermediate router commits the requested bandwidth to the packet flow. From the point-of-view of this intermediate router, the packet flow is now in progress. Each PATH and RESV message received by the intermediate router, and destination, resets the refresh timer. Periodic UPDATE messages are generated by the source and received by each router. Each router, upon processing of UPDATE messages, resets its refresh timer and propagates the UPDATE message. A packet flow is terminated, the connection torn-down, and the reserved resources released by an intermediate router (or the destination) when either an explicit TEAR-DOWN message generated by the source or destination is received, or when a refresh timer expires.

RSVP facilitates exchange of resource reservation information among routers in the packet network and is a soft-state protocol which relies upon periodic refresh message requests (UPDATE messages) to maintain router state information. Refresh messages that are not sent or processed within the period cause the established packet flow to be terminated. The periodic refresh messages, and consequent soft state information in the routers, permit the RSVP protocol to operate robustly in the presence of packet flow route changes and lost signaling messages, without requiring explicit messages to terminate the packet flow. Soft state protocols, such as RSVP, allow packet networks to provide services comparable to those in virtual circuit networks with explicit connection establishment and termination.

However, the processing section of each router must process the periodic refresh messages. Router processing load increases with the number of established RSVP packet flows passing through the router, even if these RSVP packet flows are not actively sending packets. The RSVP message load offered to the processing section comprises i) message requests due to RSVP reservation connection establishment and termination and ii) message requests due to refresh messages generated by established RSVP packet flows. Even though refresh messages consume a relatively small capacity of the processing section, the offered processor load due to refresh messages increases as the number of in-progress RSVP packet flows increases through the router. Even if an "adequate" control processor with capacity determined by traffic engineering rules is employed in the processing section, temporary overloading of the control processor may occur. Such temporary overloading may result from a "mass call-in" that generates a relatively large number of new reservation message requests for connection establishment or termination within a relatively short period of time. In addition, the control processor does not necessarily process RSVP message requests alone, but may also handle other routing tasks. With a burst of routing instabilities, and recalculations of network routes, for example, these routing tasks may require considerable portion of available processing capacity, causing a bottleneck in processing of RSVP message requests. Consequently, reservation blocking is possible even though link capacity may be available.

Appropriate scheduling of message processing by the processing section may be used to maximally utilize the link capacity within the constraints of the available processing resources. For example, consider the case when the link-utilization is relatively high. Processing PATH or RESV messages before processing TEAR-DOWN messages is not advantageous since available bandwidth is unlikely to satisfy these new requests. Therefore, for the high-link utilization case, processing TEAR-DOWN messages in the router's queue before processing PATH messages is beneficial. However, routers of the prior art employ scheduling that is not link sate dependant, such as First In First Out (FIFO) processing. Similarly, for the case when link-utilization is relatively low, processing of TEAR-DOWN messages may be deferred, allowing scarce processing

resources to process new PATH and RESV messages. However, this deference is similarly not adopted by FIFO scheduling. Furthermore, giving priority to UPDATE message processing is desirable since deferment of UPDATE message processing may result in expiration of the refresh timer, and so terminate the packet flow, in the router or in downstream routers.

## SUMMARY OF THE INVENTION

The present invention relates to allocation of processing capacity to processing control messages of a router in a packet network. A link utilization value of a link coupled to the router is monitored, and a message request size and a corresponding weight for at least one class of control messages are calculated. Each weight is calculated based on the link utilization value and the message request size of each class. A portion of the processing capacity of the router is allocated for each class of control messages based on the corresponding weight of the class.

## BRIEF DESCRIPTION OF THE DRAWINGS

Other aspects, features, and advantages of the present invention will become more fully apparent from the following detailed description, the appended claims, and the accompanying drawings in which:

FIG. **1** shows block diagram of a packet network employing a round-robin scheduling method with adaptive, weighting assignment in accordance with the present invention;

FIG. **2** shows a block diagram of an exemplary embodiment of a processing section of a router employing a scheduling method in accordance with the present invention;

FIG. **3** shows a flow chart of an algorithm implementing a round-robin scheduling method with adaptive weighting assignment employed by a scheduler of FIG. 2;

FIG. **4A** shows a number of flows for relatively low processor load without router refresh timer expiration in accordance with a scheduling algorithm of the prior art;

FIG. **4B** shows a number of flows for relatively high processor load without router refresh timer expiration in accordance with a scheduling algorithm of the prior art;

FIG. **5A** shows a number of flows for relatively low processor load with router refresh timer expiration in accordance with a scheduling algorithm of the prior art;

FIG. **5B** shows a number of flows for relatively high processor load with router refresh timer expiration in accordance with a scheduling algorithm of the prior art;

FIG. **6** shows simulation results of a moderately loaded processor employing an adaptive scheduling method in accordance with an exemplary embodiment of the present invention; and

FIG. **7** shows simulation results for the system of FIG. **6** modified to allocate a relatively high portion of processing resources to route processing.

## DETAILED DESCRIPTION

FIG. **1** shows block diagram of a packet network **100** employing a round-robin scheduling method with adaptive weighting assignment in accordance with the present invention. The packet network **100** includes a source **102** in communication with destinations **108** and **109** through routers **103, 104** and **105**. The packet network **100** employs a reservation-based protocol, such as RSVP, to allocate packet flows from the source **102** to one or more of the destinations **108** and **109** through routers **103, 104**, and **105**. The routers **103, 104** and **105** and destinations **108** and **109** employ scheduling of processing capacity of corresponding processing sections allocated to message requests of the reservation-based protocol. For the following description of exemplary embodiments, the packet network **100** employs the reservation-based protocol RSVP, although the present invention is not so limited.

A connection that establishes packet flows between source **102** and at least one destination **108** and **109** may be set up by message requests in the following manner. The source **102** desires to establish a packet flow with, for example, destination **108**. The source **102** generates a PATH message requesting, for example, a connection having a specified amount of bandwidth. The PATH message is routed through the network by routers **104** and **105**, for example, to destination **108**. Before propagating the PATH message, each router **104** and **105** checks if sufficient requested bandwidth resources are available. If the requested resources are available, router **104** first establishes a soft state for the packet flow indicated by this request and then propagates the PATH message to router **105**. Routers **104** and **105** start respective refresh timers for the packet flows that cause termination of the packet flow when no RESV or UPDATE messages are received before the refresh timer expires.

When the PATH message reaches destination **108**, destination **108** sends back a RESV message through the routers **105** and **104** to the source **102**. The RESV message may include, for example, a message request for bandwidth reservation that may be different from the bandwidth reservation message request of the PATH message. When routers **104** and **105** receive the RESV message, each router **104** and **105** commits the requested bandwidth to the packet flow, if available. Once the RESV message reaches source **102**, the connection is established. Each PATH and RESV message received by routers **104** and **105** resets the corresponding refresh timer. The connection is maintained by periodic UPDATE messages generated by the source **102** and destination **108**. Each router **104** and **105**, upon processing of an UPDATE message, resets the corresponding refresh timer and propagates the UPDATE message. The connection may either be terminated when an explicit TEAR-DOWN message generated by the source **102** or destination **108** is received, or when the refresh timer of the router **104** or **105** expires.

FIG. **2** shows a block diagram of a processing section **200** of a router, such as routers **103–105**, employing a round-robin scheduling method with adaptive weighting assignment in accordance with the present invention. The processing section **200** includes controller **202**, message-routing processor **204**, scheduler **206** having timing section **208**, and packet classifier **210**. Further included is the input queue **212** having receive queues **220** for each transmission line terminated by the router, and output buffer **214** having transmit buffers **222** for each output transmission line of the router. Each transmission line supports message or other logical traffic for one or more connections, which support may be defined as a link.

Packet classifier **210** of processing section **200** may be employed for processing of control or other system signaling type messages. Packet classifier **210** may further be part of a packet classifier processing module that also classifies data packets for traffic routing purposes. As described below, the term "packet" may indicate a control message, but may also be a portion of a control message, since control messages may be formed from several packets.

Packets received by the router are stored in input queue **212**. Packet classifier **210** monitors each packet of the input

Control / MS

queue **212**, applying, for example, a packet-filter to each packet to determine the type of packet to identify control messages: PATH, RESV, UPDATE and TEAR-DOWN messages, for example, are identified as control messages by the packet classifier **210** and reported to the scheduler **206** and controller **202**. Controller **202** processes each control message based on, for example, header information and or/message type to determine how to process the control message to establish, maintain or tear-down a connection. Message-routing processor **204**, in accordance with signals of controller **202**, transfers packets stored in the input queue **212** to a corresponding output transmission line.

Scheduler **206**, upon receiving notice of a PATH, RESV, UPDATE and TEAR-DOWN message from packet classifier **210**, begins a corresponding counter of timing section **208**. This counter may be provided as, for example, a refresh timer for a corresponding packet flow. If the counter of timing section **208** expires, an action, such as dropping the corresponding packet of the input queue **220**, is performed. The scheduler **206** also allocates processing capacity of controller **202** with a round-robin scheduling method with adaptive weighting assignment in accordance with the present invention. Briefly, the PATH, RESV, UPDATE and TEAR-DOWN messages are each a priori assigned to a class. Scheduler **206** then allocates portions of the processing capacity of the processing section **200** to each of these classes based on link utilization. The PATH, RESV, UPDATE and TEAR-DOWN messages are processed by controller **202** in accordance with the allocated processing capacity and, hence, subsequently routed by message-routing processor **204**.

The weights are calculated by a processor, which may be a processor of the controller **202** or of the scheduler **206**. For convenience, the processor of the scheduler **206** of the exemplary embodiment calculates the weights at predetermined intervals in time and also allocates processing capacity, although the present invention is not so limited. The processor may also calculate average message request size, which may be defined as the average requested quality of service (QoS) metric. The QoS metric may be an average reserved bandwidth requested by the RSVP signaling messages, but other QoS metrics may be employed that may be related to link bandwidth. For example, minimum bandwidth, transmission delay, or probability of lost packet. The method by which weights are calculated is described subsequently with respect to FIG. **3**.

To allocate portions of processing capacity for message processing by the controller **202**, a monitoring circuit to monitor link utilization may be included in the router. Such monitoring circuit may be included in the controller **202**, scheduler **206** or input queue **212**, However, the function of this circuit may be distributed in whole or in part in other circuitry within the router (e.g., in transmission line termination cards not shown in FIG. **2**). For the exemplary embodiment of FIG. **2**, the monitoring circuit may be included in the scheduler **206**. The scheduler **206** desirably monitors link utilization as, for example, traffic, the fraction of the link capacity in use. Such monitoring may include, for example, determining an average number of PATH, RESV, UPDATE and TEAR-DOWN messages received, number of established connections, average packet length, or average time in the receive queue **220**. Alternatively, message-routing processor **204** and controller **202** may monitor link utilization of each transmission line.

The round-robin scheduling method with adaptive weighting assignment as applied to the PATH, RESV, UPDATE and TEAR-DOWN message processing is now described. To simplify the following description of the preferred embodiment, refresh timers are set to fixed values that are at least an order of magnitude larger than typical round-trip times. As would be apparent to one skilled in the art, however, time adaptation mechanisms may be employed for the refresh timers. An initial model for the processing section **200** of each router **103–105** may be as follows. The processing section **200** services each logical interface of a transmission line. RSVP signaling message requests arriving over the logical interface are provided to the processing section **200** and placed in input queue **212** for further processing. A service discipline for the queue is FIFO with no distinctions made between the different message types except for weighted scheduling of the RSVP signaling message requests in accordance with the present invention.

Instead of FIFO processing of RSVP messages, weighted scheduling in accordance with the present invention processes each message type with assigned portions of processing capacity based on a priori traffic statistics, or link utilization. Weighted scheduling of the processing section **200** may be defined as an allocation of predetermined amount, or percentage, (the "weight") of overall processing capacity of the processing section **200** to a message class. This processing capacity may be the processing capacity of controller **202**, but other schemes may be employed. For example, signaling message requests may be classified into just three classes: PATH & RESV messages, UPDATE messages and TEAR-DOWN messages. Messages of each class are weighted by allocating the processing capacity (e.g., percent of processing time of the processor section) to process messages of each class in FIFO manner. "Round-robin" may be defined as switching the processing by the controller between the classes (i.e., switching the message processing) in a predetermined, cyclic order.

An immediate disadvantage of using fixed weights is the difficulty of choosing an appropriate weight for processing of UPDATE messages. The UPDATE message traffic increases in proportion to the number of RSVP packet flows already established. Moreover, if UPDATE messages are lost due to insufficient assigned weight, then existing flows may be unnecessarily torn down. Giving priority to UPDATE messages and round-robin scheduling with fixed weight processing amongst the other classes may improve performance. Round-robin scheduling with fixed weight processing, however, has further disadvantages. For example, if the message load (traffic) of UPDATE messages is very high due to a very large number of established packet flows, then the TEAR-DOWN messages may not be processed adequately. Hence, packet flows that should be torn down may last longer than necessary, increasing link utilization while preventing new packet flows from having reservation requests processed, and so connections established. Furthermore, the fixed-weight round-robin method does not account for link utilization, even though knowledge of link utilization may be used to increase the probability that message requests will be processed in a satisfactory manner.

In accordance with the present invention, round-robin scheduling with adaptive weighting assignment employs knowledge of link utilization to increase performance of message processing of each link, under varying traffic conditions. For simplicity, scheduling for three classes of service is described, although the present invention is not so limited. PATH & RESV messages are assigned to a first class, UPDATE messages are assigned to a second class, and TEAR-DOWN messages are assigned to a third class. Corresponding weights may be denoted as $w_{PR}$ for PATH &

RESV messages, $w_{UD}$ for UPDATE messages, and $w_{TD}$ for TEAR-DOWN messages. As described previously, when the counter of timing section **208** expires, an interrupt is generated for the controller **202** to terminate a respective packet flow. High priority may be assigned to these interrupts and these interrupts do not queue as TEAR-DOWN messages.

A single level scheme of round-robin scheduling with adaptive weighting assignment is now described for a case when, for example, the input queue **212** employs a single receive queue **220** of a single link (FIG. **2**). An average reservation message request size, expressed in fraction of link capacity, for PATH & RESV messages is defined as $PR_{ave}$ bits/sec. As defined herein, "size" may be the average size or amount of bandwidth, or other form of capacity (related to bandwidth) requested by the messages. The average reservation message request size $PR_{ave}$ may be computed as known in the art with an exponential smoothing model and with a forget factor $\alpha_{PR}$ selected to track message request size over time-scales of the order of a few hundred packet inter-arrival times. Similarly, an average termination (or "tear-down") message request size, expressed in fraction of link capacity, for TEAR-DOWN messages is defined as $TD_{ave}$. The value for $TD_{ave}$ may be similarly computed with a corresponding forget factor $\alpha_{TD}$. Link utilization, or utilized link capacity, is denoted as Ubits/sec, and $n_{PR}$ is defined as $(1-[U/PR_{ave}])$ and $n_{TD}$ is defined as $(U/TD_{ave})$. The weight $w_{PR}$ for processing PATH and RESV messages is now calculated as in equation (1)

$$w_{PR} = \frac{n_{PR}}{(n_{PR} + n_{TD})} \tag{1}$$

Similarly, the weight $w_{TD}$ for processing TEAR DOWN messages is calculated as in equation (2)

$$w_{TD} = \frac{n_{TD}}{(n_{PR} + n_{TD})} \tag{2}$$

Although equations (1) and (2) are shown with processing time for the three message types being the same, as would be apparent to one skilled in the art, processing time may be different in practice. However, the weights calculated from equations (1) and (2) may be scaled proportionally to the processing times.

The weights that are calculated in the equations (1) and (2) do not include a factor for the arrival rates of packets into the input queue **212**. If it is desired to account for the arrival rates to input queue **212**, then average queue length may be employed as the factor. Weights may be computed by adding the average queue length to $n_{PR}$ for the PATH & RESV message class, and then by adding the average queue length to $n_{TD}$ for the TEAR DOWN message class. Weights assigned to the classes are then computed as in equations (1) and (2).

The weight $w_{UD}$ for processing of UPDATE messages may be calculated in a similar manner to that given in equations (1) and (2) by defining an average update message size for UPDATE messages as $UD_{ave}$ forget factor $\alpha_{UD}$, $n_{UD}$ as $U/UD_{ave}$ and modifying equations (1) and (2). However, a preferred embodiment may set the weight of UPDATE message processing based on current link utilization of established packet flows. Since the UPDATE message traffic increases in proportion to the number of RSVP packet flows already established, the weight $w_U$ for processing of UPDATE messages may be adaptively varied based on the number of established RSVP packet flows in progress for the link. Therefore, the weight $w_{UD}$ may be as given in equation (3):

$$W_{UD}=\xi(\%\text{ of U for established RSVP packet flows})+C \tag{3}$$

In equation (3), $\xi$ is a scaling factor that may be experimentally optimized, and C is a constant to account for arrival rates based on queue length. Fixed priority to UPDATE messages and round-robin scheduling with adaptive weighting assignment amongst the other classes may also be employed.

As given by equations (1) and (2), a weight assigned to PATH and RESV messages is increased when U is small. The value for U may be small, for example, when reservation message request sizes are small and when the link utilization is low. Therefore, for low utilization, a scheduling method assigning weights in accordance with the present invention increases the rate at which PATH and RESV messages are processed. This increase in rate occurs since delaying processing of TEAR-DOWN messages does not affect link utilization. Similarly, when the value for U is large, the link utilization is very high and the scheduling system processes TEAR-DOWN messages at a higher rate, thereby decreasing the probability that the next PATH or RESV message is blocked.

In accordance with another exemplary embodiment of the present invention, processor scheduling for multiple links corresponding to multiple receive queues **220** of the input queue **212** shown in FIG. **2** may be employed. Processor scheduling for the case of multiple links employs a double-level hierarchical scheme of round-robin scheduling with adaptive weighting assignment instead of the single-level scheme such as described with respect to equations (1)–(3). At a high level, a super-class is defined for each link and processing capacity allocated to each link based on the corresponding weight of the super-class At a low level for each super-class of a link, the same message class definitions and weights are employed as in the single-level scheme described previously.

Linksj={1,2, i, . . . , n} (j, i, and n each an integer) are associated with the processing section **200**. Weights for the lower classes associated with each super-class are calculated in the same manner as described above with respect to equations (1) and (2). The weights for the low level class of link i are denoted by $w^i_{PR}$ and $w^i_{TD}$. For each super-class of link i, the super-class weight $W_i$ is computed as in equation (4)

$$W_i = \frac{w^i_{PR} + w^i_{TD}}{\sum_{j=1}^{n} w^j_{PR} + \sum_{j=1}^{n} w^j_{TD}} \tag{4}$$

The low level weights $w^i_{UD}$ for UPDATE message processing of each link i are determined as described above with respect to equation (3). However, the low level class weights $w^i_{UD}$ for UPDATE messages may also be calculated in a similar manner to that described for equations (1) and (2) described previously. However, for the double-level scheme the update weights $w^i_{UD}$ should then be included in equation (4) for the super-class weight $W_i$.

For this exemplary embodiment employing a double-level scheme, when a link is highly utilized (experiencing a high blocking), or a link is lightly utilized (wasting bandwidth which could be used if the processor were not a bottleneck) then one of the assigned lower-class weights of the link is high. If other links are not in these extreme situations, then the assigned weights of the other links are not as high. Hence, the super-class weights tend to give a higher share of the processor to links which are at the extremes of utilization and have a backlog of messages to be processed. This weighting assignment enforces a fairness of processing

allocation between the links. Note that if one link has a large number of reservation message requests with small size while another link has a small number of reservation message requests with large size, then the former link gets a higher super-class weight. Also, link speeds may differ since, during weight computation, link speeds and message request sizes may be normalized.

Alternative embodiments of the present invention may employ any number of modifications to the adaptive determination of weights to further account for characteristics of the packet network. For example, queue content may be included and so this information may be employed to bias the scheduling method according to message classes with longer queues, or weights may be determined only according to the queue-length of each message class. Further, weights may be multiplied by the normalized work accumulated in the corresponding message queues of the receive queues **220**.

FIG. **3** shows a flow chart of algorithm for implementing the round-robin scheduling method with adaptive weighting assignment in accordance with the present invention employed by the scheduler **206** of a router. The flow chart as shown in FIG. **3** is exemplary only. As would be apparent to one skilled in the art, the basic steps may be augmented, or the steps separately implemented in two or more processing sections of the router. In addition, the flow chart of FIG. **3** does not show the effect of expiration of a refresh timer, which effect may be implemented by employing an interrupt when the refresh timer expires for immediate processing of the connection termination.

Referring to FIG. **3**, first, at step **301**, the scheduler algorithm determines whether weights for the classes should be updated. If so, the scheduler algorithm moves to step **310**; otherwise, the scheduler algorithm moves to step **302** using, for example, weights for the assigned classes previously determined in step **311** (described subsequently). This test of step **301** may be employed in a manner such that the weight assignment method adaptively changes the weights over a reasonably short time, but also occurs relatively infrequently so as to not burden the controller or other processor of processing section **200**.

If, at step **301**, the scheduler determines that the weights should be updated, then, at step **310**, link utilization measurements are retrieved for the link or links, the individual classes of each link, and/or, if employed, the super-classes of the links. Next, at step **311**, the weights are calculated in a manner similar to that of the exemplary calculations of equations (1)–(4), and then the algorithm moves from step **311** to step **302**.

At step **302** the scheduler algorithm determines processing capacity allocated to processing messages for each of the assigned classes (e.g., PATH & RESV, UPDATE, and TEAR-DOWN messages) based on the corresponding weights. Then, the scheduler algorithm moves to step **303** to process messages of the first class.

At step **303**, the processing section processes messages of the first class in the receive queue **220** for a portion of allocated processing capacity based on the calculated weight for the first class. For example, PATH and RESV messages may be processed. The allocated portion may be a portion of the total processing capacity as measured in, for example, processor cycles, time, number of packets, or other measure of processing as known in the art. Once the allocated portion is exhausted, the scheduler algorithm moves to step **304**.

At step **304**, the processing section processes messages of the second class in the receive queue **220** for a portion of allocated processing capacity based on the calculated weight for the second class. For example, UPDATE messages may be processed. Once the allocated portion is exhausted, the scheduler algorithm moves to step **305**.

At step **305**, the processing section processes messages of the third class in the receive queue **220** for a portion of allocated processing capacity based on the calculated weight for the third class. For example, TEAR-DOWN messages may be processed. Once the allocated portion is exhausted, the scheduler algorithm moves to step **306**.

At step **306**, once the messages of the last class (e.g., TEAR-DOWN messages) are processed, the algorithm processes other messages or performs other types of packet network processing during the remaining portion of allocated processing capacity. Then, when the remaining portion is exhausted, the scheduling algorithm returns from step **306** to step **301**.

Exemplary embodiments of the present invention may be simulated and compared with a simple and useful FIFO scheduling method of the prior art. For the exemplary simulations described below with respect to FIGS. **4A–7**, a large number of sources and destinations exchange RSVP messages, with the characteristics of each reservation request varying. Other tasks that the processing section may perform, such as routing table recalculations, are accounted for in reserved processing capacity. The relative service times for RSVP message processing of the simulations were the same as those measured in existing network distributions of RSVP software.

FIGS. **4A** and **4B** show a number of packet flows for low and high message processing load cases, respectively, presented to the processing section of a router in accordance with a FIFO scheduling method of the prior art having a disabled refresh timer. FIFO processing for PATH, RESV, and UPDATE messages is employed in the simulations of FIGS. **4A** and **4B**, and explicit TEAR-DOWN messages are given absolute priority. Not processing TEAR-DOWN messages in a FIFO manner and giving absolute priority releases link capacity (decreases link utilization) and reduces the chance of other requests being blocked. Message request size may be the same for all reservation message classes (i.e., models a scenario with many flows of same type). Connection or call holding time is defined as 300 seconds, message processing time is on the order of 100 ms, and TEAR-DOWN message processing time is two or three times higher than the PATH and RESV message processing times.

Link utilization, expressed in number of packet flows since message request size is defined as a constant size, is shown in FIGS. **4A** and **4B** as a function of time for the two cases of processor utilization (low processor load and high processor load). For the high processor load case shown in FIG. **4B**, the load offered is below capacity without TEAR-DOWN messages. Once TEAR-DOWN messages are included the load often exceeds available processing capacity. For the cases of the exemplary embodiment of FIGS. **4A** and **4B**, the refresh timer is disabled such that it does not expire, and the link utilization of the router is shown to be unsatisfactory.

Referring to FIG. **4A**, initially, when the number of packet flows is not very large, the processing load is low enough that many flows are successfully established. Note that to establish a packet flow, both its PATH and RESV message must be successfully processed. As the number of flows increases, the UPDATE message traffic increases proportionally and, hence, increases processor load. The number of established packet flows continues to increase. In addition, some of the established packet flows start generating TEAR-

DOWN messages, since their holding times have elapsed. Because TEAR-DOWN messages are given priority, the number of packet flows in progress decreases.

When the number of packet flows has decreased sufficiently, the arrival rate of TEAR-DOWN messages decreases sufficiently such that new packet flows are established at a faster rate than the rate at which packet flows are terminated or tom down. Hence, the number of packet flows increases again. After a delay, equal to the message holding time, TEAR-DOWN messages are generated again and the number of flows (and hence link utilization) goes down. This oscillation in packet flow of FIG. 4A is more pronounced at high message load, such as is shown in the simulation results of FIG. 4B.

FIGS. 5A and 5B show a number of packet flows for low and high message processing load cases, respectively, presented to the processing section of a router in accordance with a FIFO scheduling method of the prior art with a refresh timer enabled. The oscillation in packet flow is similar to that shown in FIGS. 4A and 4B. Extreme oscillation in packet flow occurs in both cases shown in FIGS. 5A and 5B. Since UPDATE messages are processed in a FIFO manner, UPDATE messages are queued and are significantly delayed, or lost, if the queue (buffers) is not large. Delaying queued UPDATE messages causes the refresh timer to expire even if set to a value an order of magnitude greater than end-to-end packet round-trip times.

FIG. 6 shows simulation results for a moderately loaded processing section employing a round-robin scheduling method with both fixed weighting assignment and adaptive weighting assignment in accordance with an exemplary embodiment of the present invention. The simulation results for the round-robin scheduling method with fixed weights are labeled FWRR, and have weights chosen to be proportional to service times. The simulation results for the round-robin scheduling method with adaptive weight assignment in accordance with the present invention are labeled AWRR. Also shown in FIG. 6 are results of the FIFO scheduling method of the prior art under moderate processing load.

For the simulation results shown in FIG. 6, and FIG. 7 described subsequently, all reservation message request sizes are between 1 and 5 kbits/s; the mean call holding time is 180 seconds; and the inter-arrival times of the message requests follow an exponential distribution. For the exemplary simulations, the greatest number of bandwidth reservation requests are for small bandwidth reservation (e.g. for audio conferences of about 64 kb/sec). In addition, some sources generate requests for much larger bandwidth reservation (e.g., for video servers or video conferencing systems).

As shown in the simulation results of FIG. 6, the bandwidth reserved for each scheduling method is plotted as a function of time. Since the processor load is moderate, the effect of scheduling is not very pronounced and the reserved bandwidth is actually somewhat higher for the FIFO scheduling method.

However, the higher reserved bandwidth of the FIFO scheduling method is because TEAR-DOWN messages are delayed and processed later. Consequently, reserved bandwidth of the FIFO scheduler is wasted because the receiver or sender initiated connection tear-down that terminates bandwidth usage by the packet flow has not been processed. Both the FWRR and, in particular, the AWRR scheduling methods reclaim the reserved bandwidth faster, and hence reserved bandwidth of the FWRR and WRR methods appears lower than that of FIFO scheduling method.

However, more spurious connection terminations occur for the FIFO scheduling method because UPDATE messages are not processed before expiration of the corresponding refresh timer. The number of spurious terminations is smaller for the FWRR scheduling method and is lowest for the AWRR scheduling method, showing the advantage of adaptive weighting assignment for scheduling of processor capacity in accordance with the present invention.

FIG. 7 shows simulation results for a simulation system similar to that of FIG. 6 but with an added high-processing load due to route processing by the router's processing section. Total processing load is, therefore, much higher even though the offered load due to RSVP messages is the same. The results of the round-robin scheduling method with fixed weights are labeled FWRR, the results of the round-robin scheduling method with adaptive weight assignment in accordance with the present invention are labeled AWRR, and the results of the FIFO scheduling method of the prior art are labeled FIFO.

When the processing load increases, the advantages of the AWRR scheduling method are shown for processing a number of flows in progress, and for reserved bandwidth. As shown in the simulation results of FIG. 7, the AWRR scheduling method is effective, for example, in extreme cases where reserved bandwidth is very low and when the reserved bandwidth reaches the maximum link bandwidth utilization. For the first case where reserved bandwidth is very low, a number of accepted reservations increases. For the second case where reserved bandwidth reaches a maximum, the number of established, or accepted, connections is maximized since reserved bandwidth of terminated connections is freed faster than with the FWRR scheduling method.

A router employing a round-robin scheduling method with adaptive weighting assignment in accordance with the present invention for processing of message classes allows for five desirable processing features when a reservation-based protocol is employed in a packet network. First, refresh messages (UPDATE messages) generally require at least a fraction of the available bandwidth, and this fixed fraction may be made an increasing function of the number of packet flows in progress. An upper bound may be determined so as to maintain a minimum bandwidth available for other message types. Second, when link utilization is low, reservation message requests for establishing a connection (e.g., PATH and RESV messages) may be assigned a higher weight since delay of TEAR-DOWN processing generally does not adversely impact request blocking. However, when link utilization is high, processing of termination messages (e.g., TEAR-DOWN messages) may be given a higher weight since processing PATH or RESV messages before processing of TEAR-DOWN messages may result in the bandwidth request fore each PATH or RESV message being denied.

Third, assigned weights may be adjusted based on the size of average recent packet flow establishment and termination requests. For example, if reservations are small and link utilization is low, then the weight assigned to RESV messages may be increased since processing of each message has a much smaller impact on the link utilization. Similarly if average recent flow establishment and termination requests sizes are large, then the weights for reservations may be scaled down. Fourth, processing time for each message type, if known, may be accounted for in assigning weights. Fifth, instantaneous queue-lengths by message type may be employed to control queue lengths in extreme situations when the link is totally over-utilized or totally under-utilized.

While the exemplary embodiments of the present invention have been described with respect to processing method,

the present invention is not so limited. As would be apparent to one skilled in the art, various functions may also be implemented in the circuits or a combination of circuits and in digital domain as processing steps in a software program of, for example, a micro-controller or general purpose computer.

It will be further understood that various changes in the details, materials, and arrangements of the parts which have been described and illustrated in order to explain the nature of this invention may be made by those skilled in the art without departing from the principle and scope of the invention as expressed in the following claims.

What is claimed is:

1. A processing section of a router for processing control messages in a packet network, the processing section comprising:

a monitoring module adapted to monitor a link utilization value of a link coupled to the router;

a processor to calculate a message request size and a corresponding weight for at least one class of control messages, each weight calculated based on the link utilization value and each message request size; and

a scheduling module adapted to allocate, for each class of control messages, a portion of the processing capacity of the processing section based on the corresponding weight of the class.

2. The invention as recited in claim 1, wherein the control messages further include an update message class of control messages for maintaining at least one established packet flow of the link, and the processor further calculates the weight for the update message class based on the number of established packet flows of the link.

3. The invention as recited in claim 2, wherein the control messages are in accordance with a reservation-based protocol, and the control messages include a first class of control messages for establishing at least one packet flow of the link and a second class of control messages for terminating at least one packet flow of the link.

4. The invention as recited in claim 2, wherein the message request size is based on an average of a requested link characteristic of the control messages.

5. The invention as recited in claim 4, wherein the requested link characteristic of the control messages is either bandwidth, transmission delay, or probability of lost packet.

6. The invention as recited in claim 1, wherein:

the monitoring module further monitors link utilization values for two or more links coupled to the router;

the processor further calculates, for each link, a message request size and corresponding weight for each class of control messages based on the link utilization of the link, the processor further adapted to calculate a super-class weight for each link; and

the scheduling module allocates the processing capacity of the processing section to each link based on the corresponding super-class weight, and allocates a portion of the processing capacity allocated to the link to each class of the link based on the corresponding weight of the class.

7. A method for allocating processing capacity to control messages received by a router in a packet network, the method comprising the steps of:

a) monitoring a link utilization value of a link coupled to the router;

b) calculating a message request size and a corresponding weight for at least one class of control messages, each weight calculated based on the link utilization value and each message request size; and

c) allocating, for each class of control messages, a portion of the processing capacity of the router based on the corresponding weight of the class.

8. The method as recited in claim 7, wherein the control messages further include an update message class of control messages for maintaining at least one established packet flow of the link, and the calculating step b) further includes the step of b1) calculating the weight for the update message class based on the number of established packet flows of the link.

9. The method as recited in claim 8, wherein, for the calculating step b), the control messages are in accordance with a reservation-based protocol, and the control messages include a first class of control messages for establishing at least one packet flow of the link and a second class of control messages for terminating at least one packet flow of the link.

10. The method as recited in claim 8, wherein, for the calculating step b), the message request size of a class is calculated based on an average of a requested link characteristic of the control messages.

11. The method as recited in claim 10, wherein, for the calculating step b), the requested link characteristic is either bandwidth, transmission delay, or probability of lost packet.

12. The method as recited in claim 7, wherein:

the monitoring step a) further includes the step of a1) monitoring link utilization values for two or more links coupled to the router;

the calculating step b) further includes the steps of b2) calculating, for each link, a message request size and corresponding weight for each class of control messages based on the link utilization of the link, and b3) calculating a super-class weight for each link; and

the allocating step c) further includes the step of c1) allocating the processing capacity to each link based on the corresponding super-class weight, and c2) allocating a portion of the processing capacity allocated to the link to each class of the link based on the corresponding weight of the class.

13. A router of an IP packet network having a processing section for processing control messages in accordance with a reservation-based protocol, the processing section comprising:

a monitoring module adapted to monitor a link utilization value of a link coupled to the router;

a processor adapted to calculate a message request size and a corresponding weight for at least one class of control messages, each weight calculated based on the link utilization value and each message request size; and

a scheduling module adapted to allocate, for each class of control messages, a portion of the processing capacity of the processing section based on the corresponding weight of the class.

14. The invention as recited in claim 13, wherein the control messages further include an update message class of control messages for maintaining at least one established packet flow of the link, and the processor further calculates the weight for the update message class based on the number of established packet flows of the link.

15. The invention as recited in claim 14, wherein:

the monitoring module further monitors link utilization values for two or more links coupled to the router;

the processor further calculates, for each link, a message request size and corresponding weight for each class of control messages based on the link utilization of the link, the processor further adapted to calculate a super-class weight for each link; and

the scheduling module allocates the processing capacity of the processing section to each link based on the corresponding super-class weight, and allocates a portion of the processing capacity allocated to the link to each class of the link based on the corresponding weight of the class.

* * * * *

US005940390A

# United States Patent [19]

Berl et al.

[11] Patent Number: 5,940,390

[45] Date of Patent: Aug. 17, 1999

[54] **MECHANISM FOR CONVEYING DATA PRIORITIZATION INFORMATION AMONG HETEROGENEOUS NODES OF A COMPUTER NETWORK**

[75] Inventors: **Steven H. Berl**, Piedmont; **Ulrica Tam**, Belmont, both of Calif.

[73] Assignee: **Cisco Technology, Inc.**, San Jose, Calif.

[21] Appl. No.: **08/833,834**

[22] Filed: **Apr. 10, 1997**

[51] **Int. Cl.**[6] ..... **G06F 13/00**

[52] **U.S. Cl.** ..... **370/389**; 370/469; 395/200.57

[58] **Field of Search** ..... 370/236, 384, 370/400, 401, 410, 412, 414, 426, 465, 469, 389, 522, 466; 395/200.57, 200.58

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,416,769 5/1995 Karol ..... 370/414

OTHER PUBLICATIONS

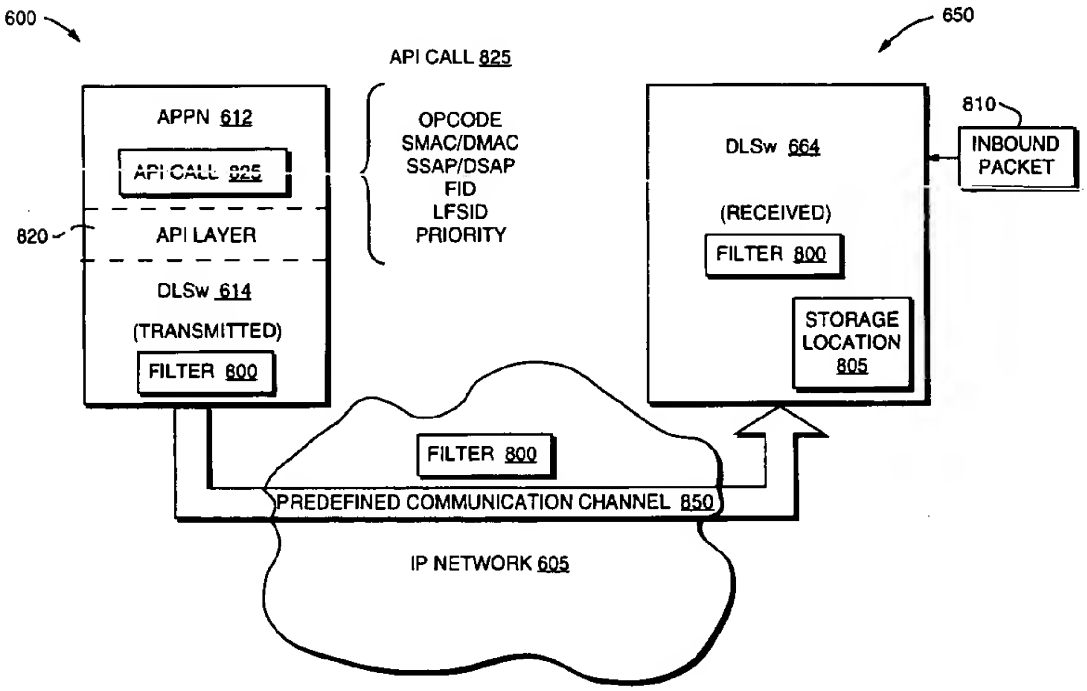
Nilausen, Jesper—APPN Networks; John Wiley & Sons, Ltd. 1994; APPN Basics, 2:11–83.

*Primary Examiner*—Chi H. Pham
*Assistant Examiner*—Maikhanh Tran
*Attorney, Agent, or Firm*—Cesari and McKenna, LLP

[57] **ABSTRACT**

A mechanism conveys information pertaining to transmission priority (TP) levels of inbound packets transmitted over a heterogeneous network from a switching node to a hybrid node of the network. The mechanism comprises a packet-recognizing filter having a format that is generated by the hybrid node and dynamically transmitted to the switching node over a predefined communication channel of the network. The filter enables the switching node to classify the inbound packets and assign them appropriate TP levels.

**21 Claims, 10 Drawing Sheets**



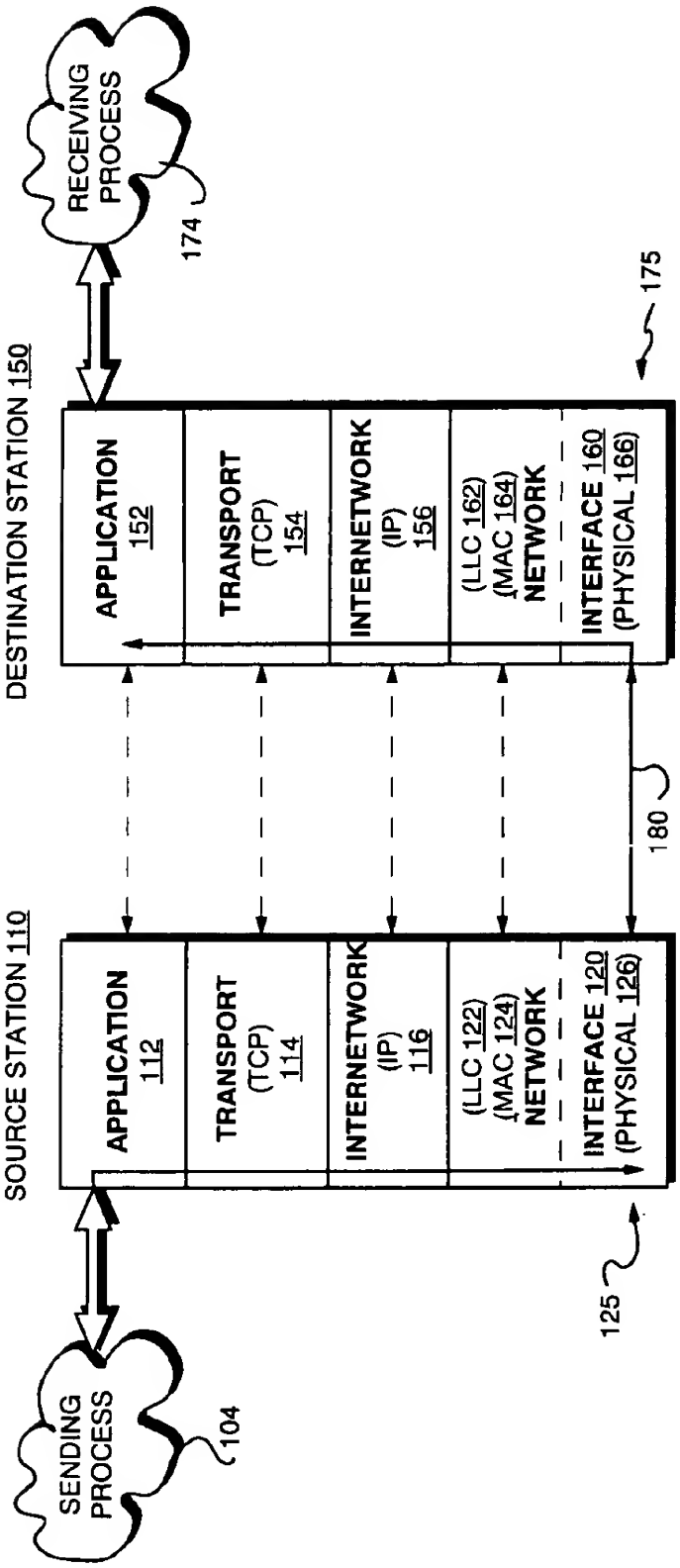



FIG. 1
(PRIOR ART)

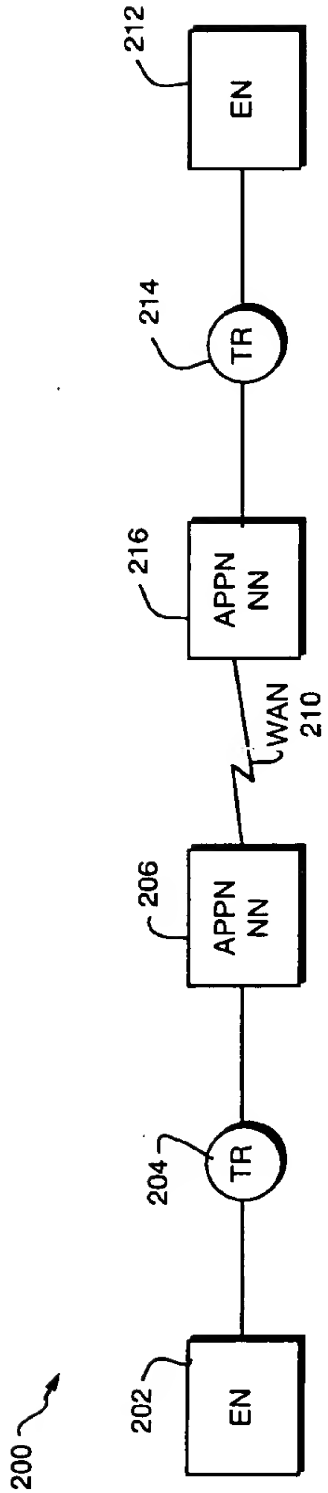



FIG. 2
(PRIOR ART)

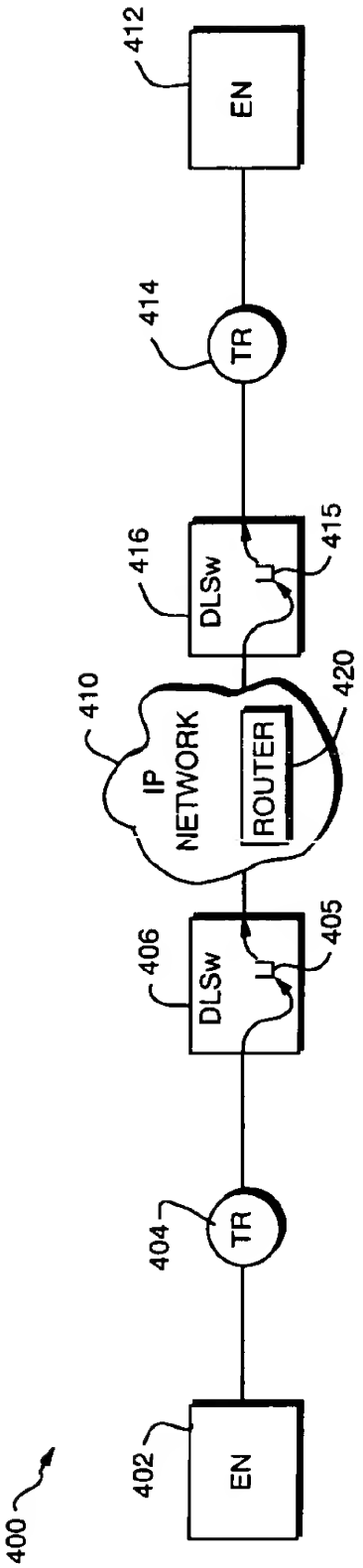



FIG. 4
(PRIOR ART)

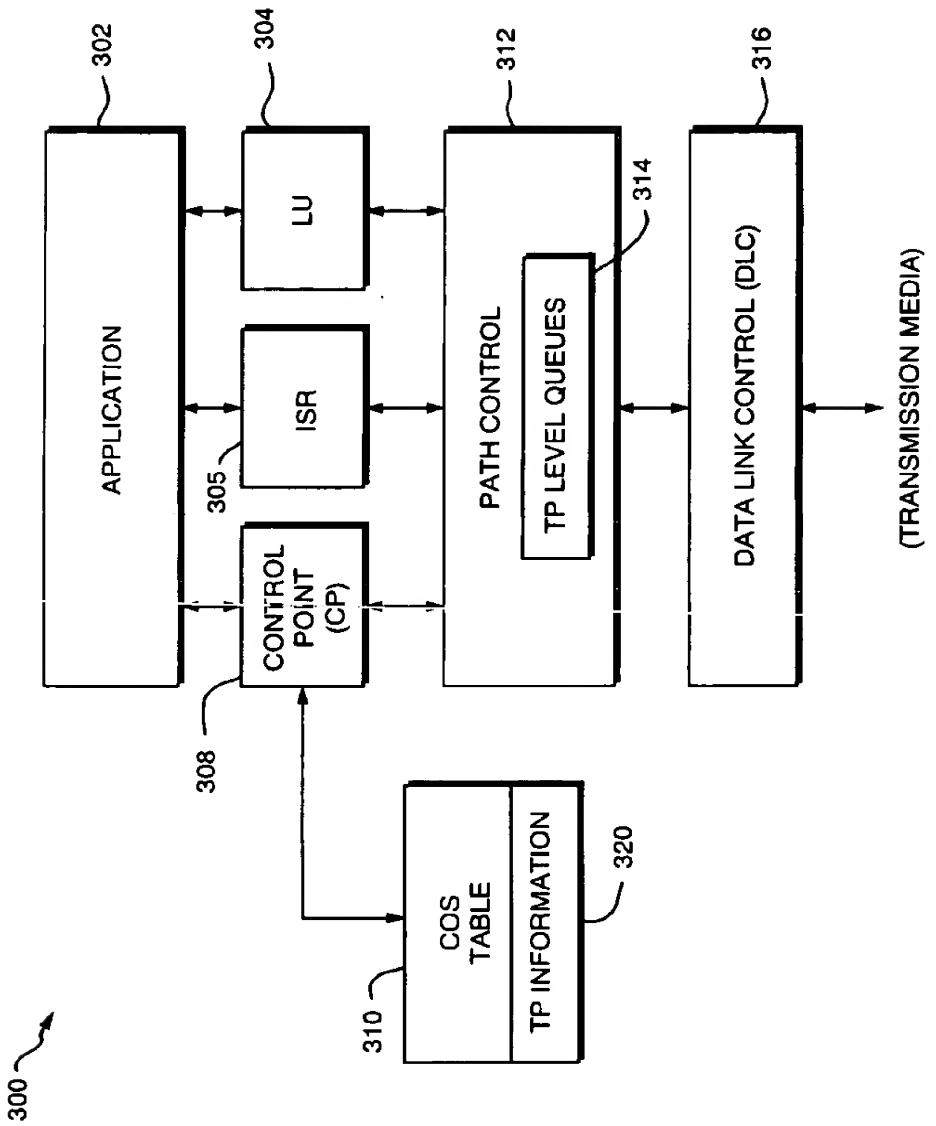



FIG. 3
(PRIOR ART)

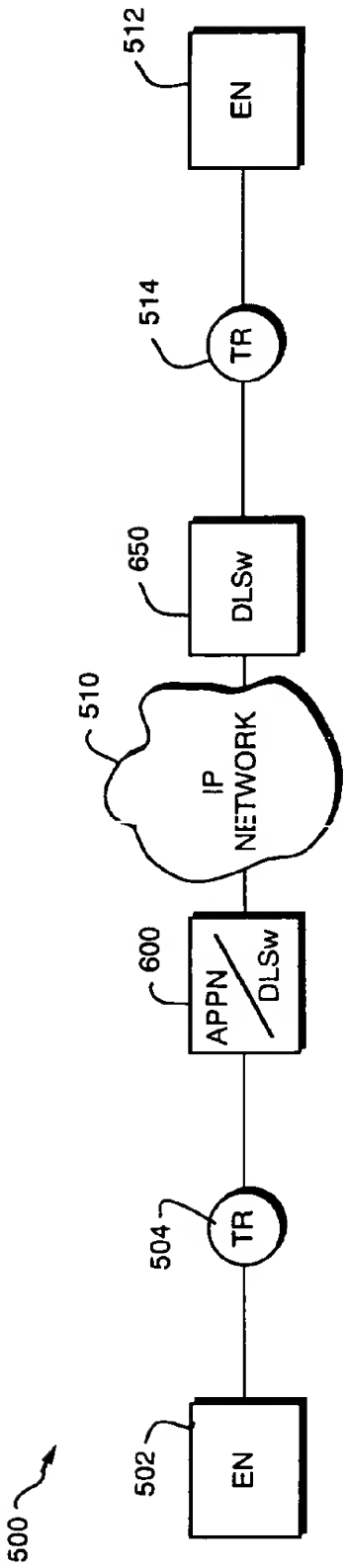

500
502
EN
504
TR
600
APPN / DLSw
510
IP NETWORK
650
DLSw
514
TR
512
EN

FIG. 5

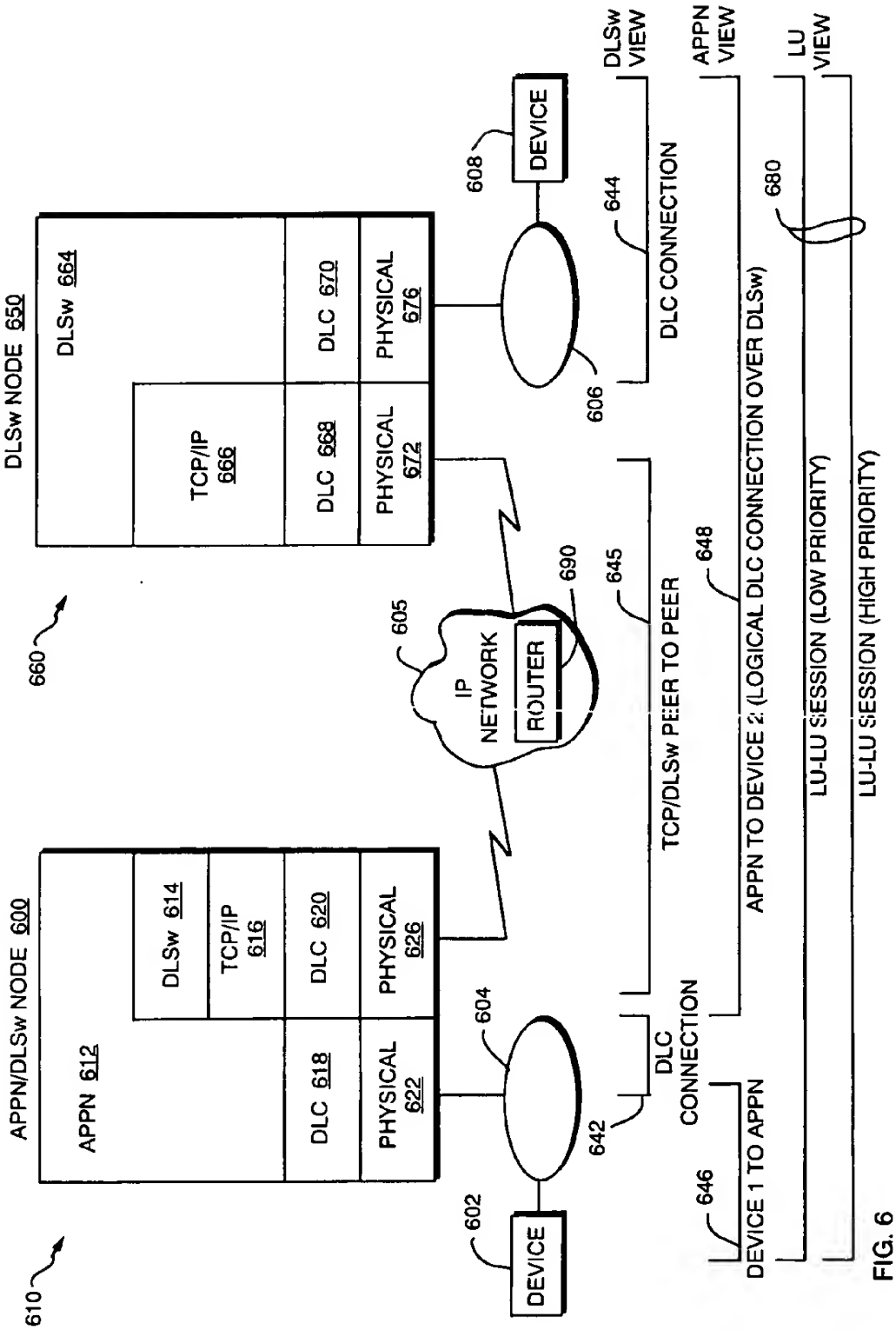

APPN/DLSw NODE 600
APPN 612
DLSw 614
TCP/IP 616
DLC 618
DLC 620
PHYSICAL 622
PHYSICAL 626
610
602
DEVICE
604
642
DLC CONNECTION
646
DEVICE 1 TO APPN
DLSw NODE 650
DLSw 664
TCP/IP 666
DLC 668
DLC 670
PHYSICAL 672
PHYSICAL 676
660
605
IP NETWORK
ROUTER
690
645
TCP/DLSw PEER TO PEER
606
644
DLC CONNECTION
608
DEVICE
DLSw VIEW
APPN VIEW
LU VIEW
648
APPN TO DEVICE 2 (LOGICAL DLC CONNECTION OVER DLSw)
680
LU-LU SESSION (LOW PRIORITY)
LU-LU SESSION (HIGH PRIORITY)

FIG. 6

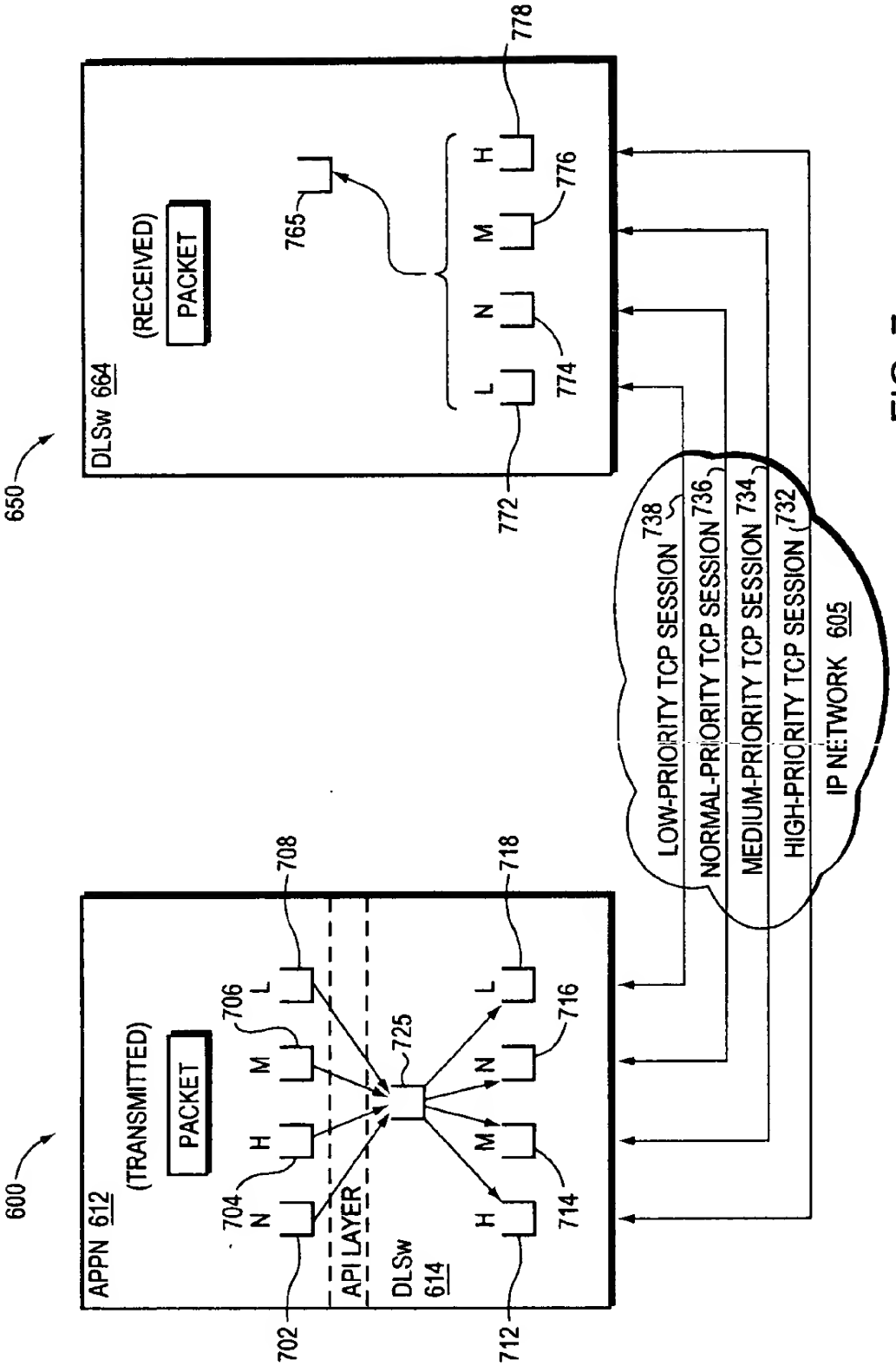

600
APPN 612
(TRANSMITTED)
PACKET
704
N
H
706
M
L
708
API LAYER
DLSw 614
725
702
712
718
L
N
M
H
716
714
LOW-PRIORITY TCP SESSION 738
NORMAL-PRIORITY TCP SESSION 736
MEDIUM-PRIORITY TCP SESSION 734
HIGH-PRIORITY TCP SESSION 732
IP NETWORK 605
650
DLSw 664
(RECEIVED)
PACKET
765
L
N
M
H
774
776
778
772

FIG. 7

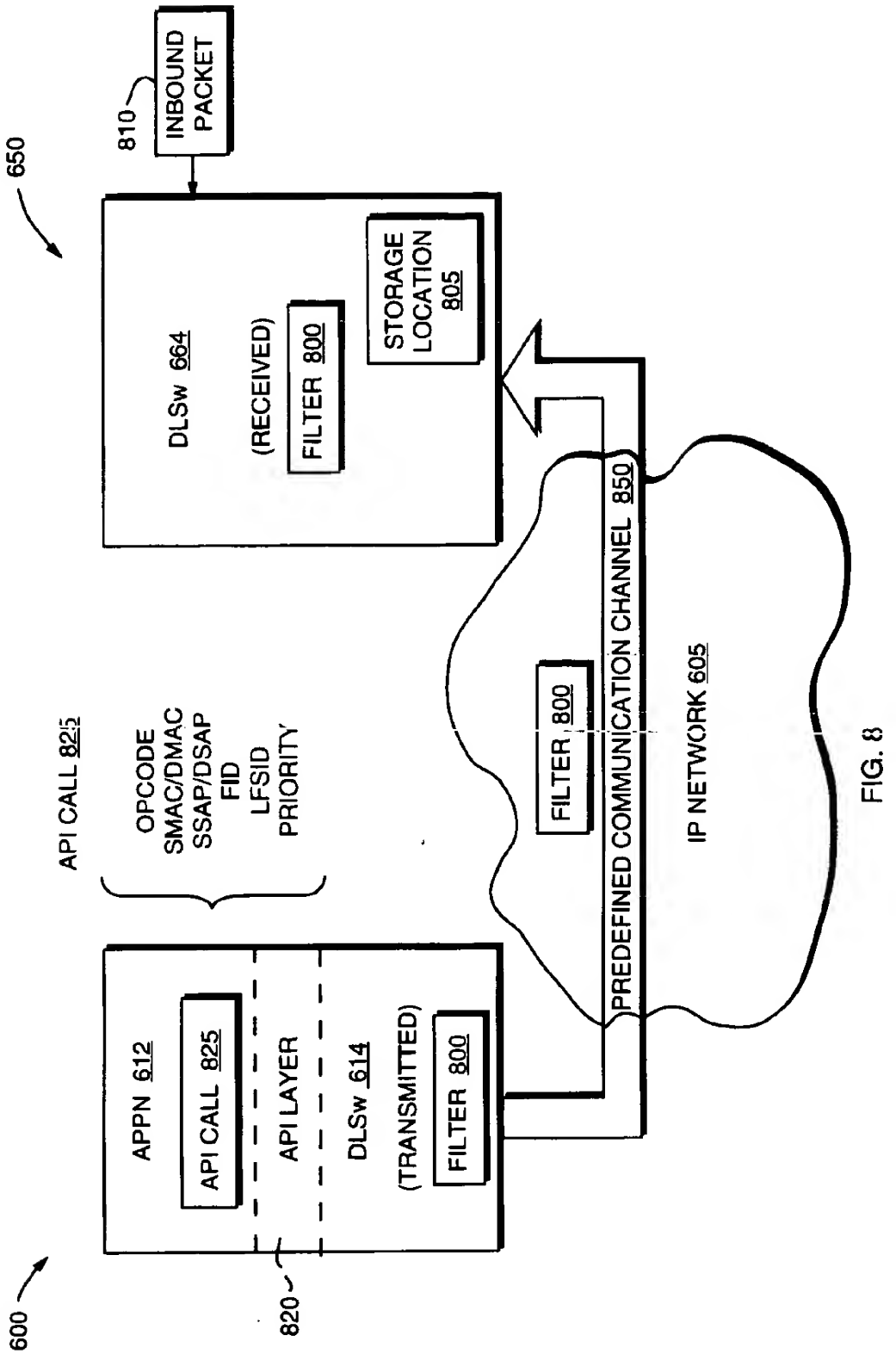

600
650
810
INBOUND PACKET
DLSw 664
(RECEIVED)
FILTER 800
STORAGE LOCATION 805
PREDEFINED COMMUNICATION CHANNEL 850
IP NETWORK 605
FILTER 800
API CALL 825
OPCODE
SMAC/DMAC
SSAP/DSAP
FID
LFSID
PRIORITY
APPN 612
API CALL 825
API LAYER
820
DLSw 614
(TRANSMITTED)
FILTER 800

FIG. 8

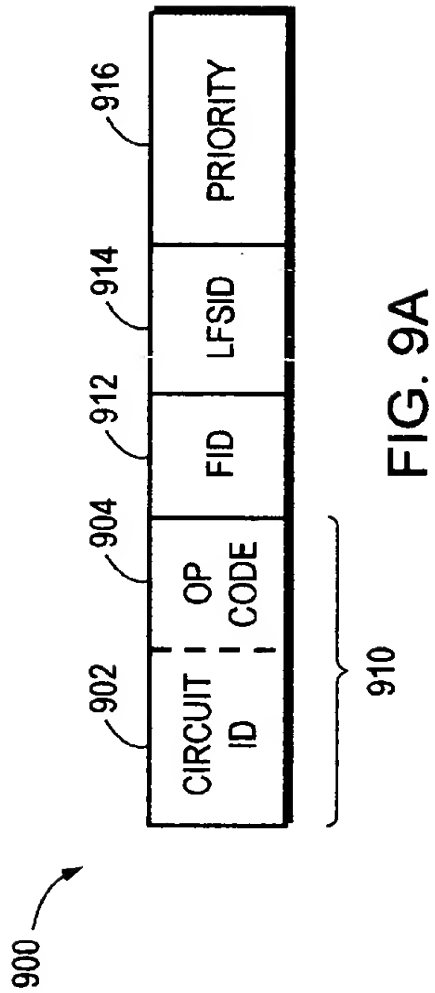

900
902
CIRCUIT ID
904
OP CODE
912
FID
914
LFSID
916
PRIORITY
910

FIG. 9A

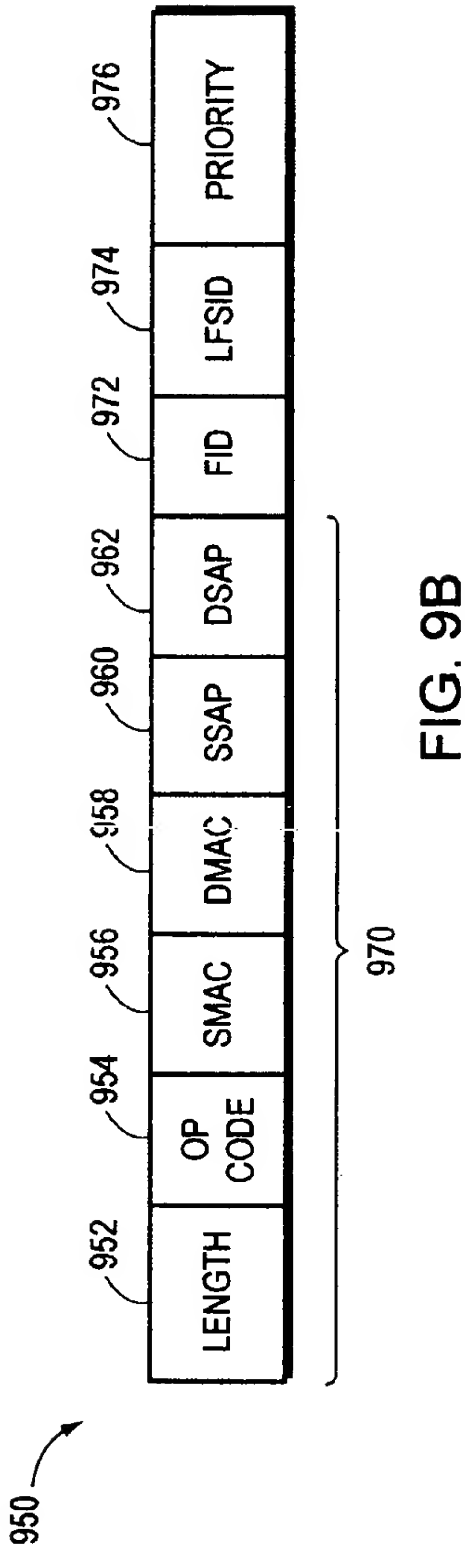
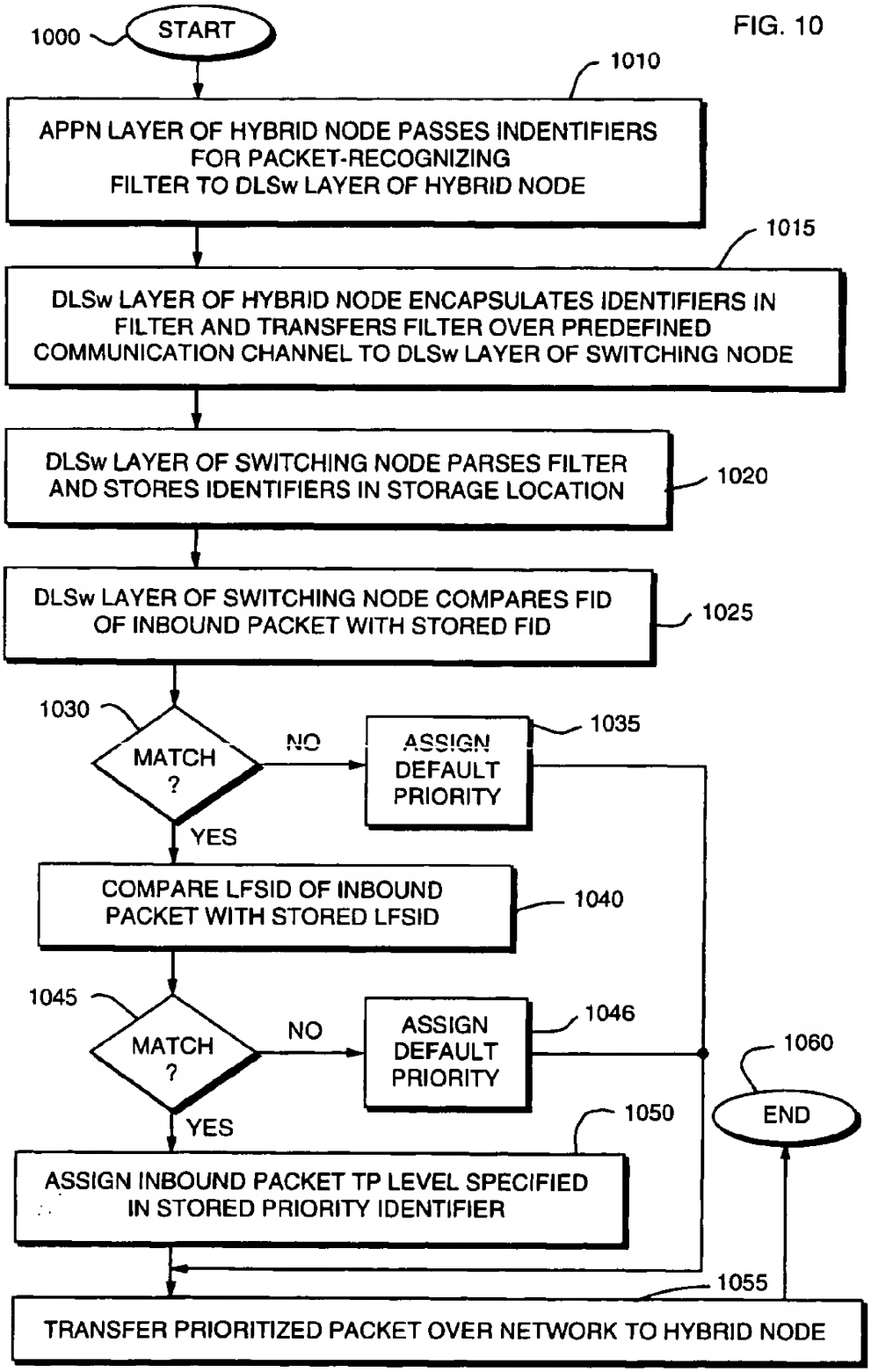

950
952
LENGTH
954
OP CODE
956
SMAC
958
DMAC
960
SSAP
962
DSAP
972
FID
974
LFSID
976
PRIORITY
970

FIG. 9B

FIG. 10


1000
START
1010
APPN LAYER OF HYBRID NODE PASSES INDENTIFIERS FOR PACKET-RECOGNIZING FILTER TO DLSw LAYER OF HYBRID NODE
1015
DLSw LAYER OF HYBRID NODE ENCAPSULATES IDENTIFIERS IN FILTER AND TRANSFERS FILTER OVER PREDEFINED COMMUNICATION CHANNEL TO DLSw LAYER OF SWITCHING NODE
DLSw LAYER OF SWITCHING NODE PARSES FILTER AND STORES IDENTIFIERS IN STORAGE LOCATION
1020
DLSw LAYER OF SWITCHING NODE COMPARES FID OF INBOUND PACKET WITH STORED FID
1025
1030
MATCH ?
NO
ASSIGN DEFAULT PRIORITY
1035
YES
COMPARE LFSID OF INBOUND PACKET WITH STORED LFSID
1040
1045
MATCH ?
NO
ASSIGN DEFAULT PRIORITY
1046
YES
1050
ASSIGN INBOUND PACKET TP LEVEL SPECIFIED IN STORED PRIORITY IDENTIFIER
1060
END
1055
TRANSFER PRIORITIZED PACKET OVER NETWORK TO HYBRID NODE

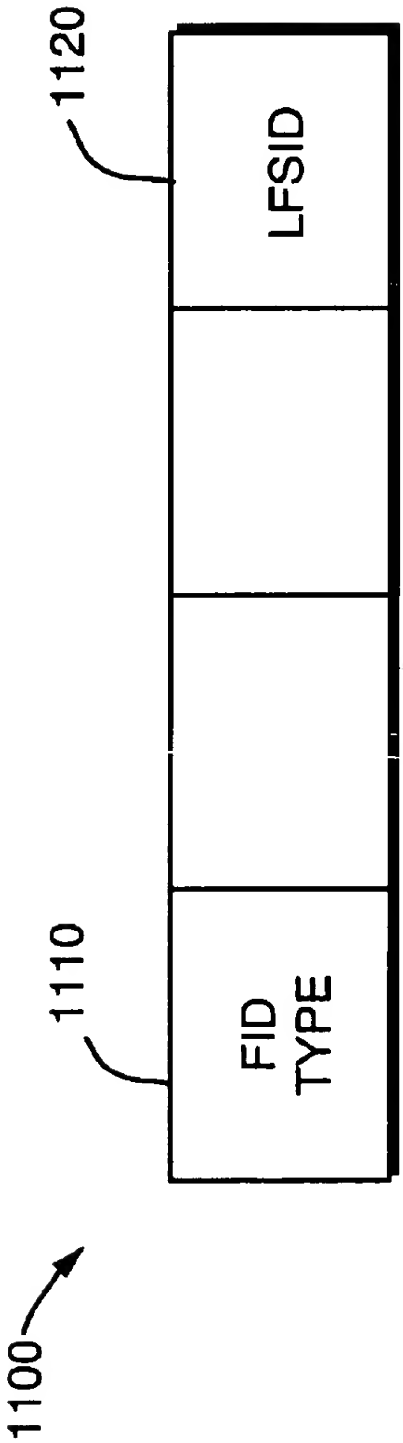



FIG. 11
(PRIOR ART)

# MECHANISM FOR CONVEYING DATA PRIORITIZATION INFORMATION AMONG HETEROGENEOUS NODES OF A COMPUTER NETWORK

## CROSS-REFERENCE TO RELATED APPLICATIONS

This invention is related to the following copending U.S. patent application:

U. S. patent application Ser. No. 08/839,435, titled TECHNIQUE FOR MAINTAINING PRIORITIZATION OF DATA TRANSFERRED AMONG HETEROGENEOUS NODES OF A COMPUTER NETWORK. U.S. patent application Ser. No. 08/833,837, titled TECHNIQUE FOR CAPTURING INFORMATION NEEDED TO IMPLEMENT TRANSMISSION PRIORITY ROUTING AMONG HETEROGENEOUS NODES OF A COMPUTER NETWORK, which applications were filed on even date herewith and assigned to the assignee of the present invention.

U.S. patent application Ser. No. 08/926,539, titled TECHNIQUE FOR REDUCING THE FLOW OF TOPOLOGY INFORMATION AMONG NODES OF A COMPUTER NETWORK, which application was filed on Sep. 10, 1997 and assigned to the assignee of the present invention.

## FIELD OF THE INVENTION

The invention relates to computer networks and, more particularly, to the distribution of packet prioritization information among stations of a computer network.

## BACKGROUND OF THE INVENTION

Data communication in a computer network involves the exchange of data between two or more entities interconnected by communication links and sub-networks. These entities are typically software programs executing on hardware computer platforms, such as end stations and intermediate stations. Examples of an intermediate station may be a router or switch which interconnects the communication links and subnetworks to enable transmission of data between the end stations. A local area network (LAN) is an example of a subnetwork that provides relatively short distance communication among the interconnected stations; in contrast, a wide area network (WAN) enables long distance communication over links provided by public or private telecommunications facilities.

Communication software executing on the end stations correlate and manage data communication with other end stations. The stations typically communicate by exchanging discrete packets or frames of data according to predefined protocols. In this context, a protocol consists of a set of rules defining how the stations interact with each other. In addition, network routing software executing on the routers allow expansion of communication to other end stations. Collectively, these hardware and software components comprise a communications network and their interconnections are defined by an underlying architecture.

Modern communications network architectures are typically organized as a series of hardware and software levels or "layers" within each station. These layers interact to format data for transfer between, e.g., a source station and a destination station communicating over the network. Specifically, predetermined services are performed on the data as it passes through each layer and the layers communicate with each other by means of the predefined protocols.



The lower layers of these architectures are generally standardized and are typically implemented in hardware and firmware, whereas the higher layers are generally implemented in the form of software running on the stations attached to the network. Examples of such communications architectures include the Systems Network Architecture (SNA) developed by Inteerational Business Machines Corporation and the Internet communications architecture.

The Internet architecture is represented by four layers which are termed, in ascending interfacing order, the network interface, internetwork, transport and application layers. These layers are arranged to form a protocol stack in each communicating station of the network. FIG. 1 illustrates a schematic block diagram of prior art Internet protocol stacks **125** and **175** used to transmit data between a source station **110** and a destination station **150**, respectively, of a network **100**. As can be seen, the stacks **125** and **175** are physically connected through a communications channel **180** at the network interface layers **120** and **160**. For ease of description, the protocol stack **125** will be described.

In general, the lower layers of the communications stack provide internetworking services and the upper layers, which are the users of these services, collectively provide common network application services. The application layer **112** provides services suitable for the different types of applications using the network, while the lower network interface layer **120** of the Internet architecture accepts industry standards defining a flexible network architecture oriented to the implementation of LANs.

Specifically, the network interface layer **120** comprises physical and data link sublayers. The physical layer **126** is concerned with the actual transmission of signals across the communication channel and defines the types of cabling, plugs and connectors used in connection with the channel. The data link layer, on the other hand, is responsible for transmission of data from one station to another and may be further divided into two sublayers: Logical Link Control (LLC **122**) and Media Access Control (MAC **124**).

The MAC sublayer **124** is primarily concerned with controlling access to the transmission medium in an orderly manner and, to that end, defines procedures by which the stations must abide in order to share the medium. In order for multiple stations to share the same medium and still uniquely identify each other, the MAC sublayer defines a hardware or data link address called a MAC address. This MAC address is unique for each station interfacing to a LAN. The LLC sublayer **122** manages communications between devices over a single link of the network and provides for environments that need connectionless or connection-oriented services at the data link layer.

Connection-oriented services at the data link layer generally involve three distinct phases: connection establishment, data transfer and connection termination. During connection establishment, a single path is established between the source and destination stations. This connection, e.g., an IEEE 802.2 LLC Type 2 or "Data Link Control" (DLC) connection as referred hereinafter, is based on the use of service access points (SAPs); a SAP is generally the address of a port or access point to a higher-level layer of a station. Once the connection has been established, data is transferred sequentially over the path and, when the DLC connection is no longer needed, the path is terminated. The details of such connection establishment and termination are well-known and, thus, will not be described herein.

The transport layer **114** and the internetwork layer **116** are substantially involved in providing predefined sets of services to aid in connecting the source station to the destination station when establishing application-to-application communication sessions. The primary network layer protocol of the Internet architecture is the Internet protocol (IP) contained within the internetwork layer **116**. IP is primarily a connectionless network protocol that provides internetwork routing, fragmentation and reassembly of datagrams and that relies on transport protocols for end-to-end reliability. An example of such a transport protocol is the Transmission Control Protocol (TCP) contained within the transport layer **114**. Notably, TCP provides connection-oriented services to the upper layer protocols of the Internet architecture. The term TCP/IP is commonly used to refer to the Internet architecture.

Data transmission over the network **100** therefore consists of generating data in, e.g., sending process **104** executing on the source station **110**, passing that data to the application layer **112** and down through the layers of the protocol stack **125**, where the data are sequentially formatted as a frame for delivery onto the channel **180** as bits. Those frame bits are then transmitted over an established connection of channel **180** to the protocol stack **175** of the destination station **150** where they are passed up that stack to a receiving process **174**. Data flow is schematically illustrated by solid arrows.

Although actual data transmission occurs vertically through the stacks, each layer is programmed as though such transmission were horizontal. That is, each layer in the source station **110** is programmed to transmit data to its corresponding layer in the destination station **150**, as schematically shown by dotted arrows. To achieve this effect, each layer of the protocol stack **125** in the source station **110** typically adds information (in the form of a header field) to the data frame generated by the sending process as the frame descends the stack. At the destination station **150**, the various encapsulated headers are stripped off one-by-one as the frame propagates up the layers of the stack **175** until it arrives at the receiving process.

SNA is a mainframe-oriented network architecture that also uses a layered approach. The services included within this architecture are generally similar to those defined in the Internet communications architecture. In a SNA network, though, applications executing on end stations typically access the network through logical units (LU) of the stations; accordingly, in a typical SNA network, a communication session connects two LUs in a LU-LU session. Activation and deactivation of such a session is addressed by Advanced Peer to Peer Networking (APPN) functions.

The APPN functions generally include session establishment and session routing within an APPN network. FIG. **2** is a schematic block diagram of a prior art APPN network **200** comprising two end stations **202, 212**, which are typically configured as end nodes (EN), coupled to token ring (TR) subnetworks **204, 214**, respectively. During session establishment, an EN (such as EN **202**) requests an optimum route for a session between two LUs; this route is calculated and conveyed to EN **202** by an intermediate station functioning as a network node server (e.g., station **206**) via a LOCATE message exchange through the network **200**. Thereafter, a "set-up" or BIND message is forwarded over the route to initiate the session. The BIND includes information pertaining to the partner LU requested for the session.

Intermediate session routing occurs when the intermediate stations **206, 216**, configured as APPN network nodes (NN), are present in a session between the two end nodes. As can be seen, the APPN network nodes are further interconnected by a WAN **210** that extends the APPN architecture throughout the network. The APPN network nodes forward packets of an LU-LU session over the calculated route between the two APPN end nodes. An APPN network node is a full-functioning APPN router node having all APPN base service capabilities, including session services functions. An APPN end node, on the other hand, is capable of performing only a subset of the functions provided by an APPN network node. APPN network and end nodes are well-known and are, for example, described in detail in *Systems Network Architecture Advanced Peer to Peer Networking Architecture Reference* IBM Doc SC30-3422 and *APPN Networks* by Jesper Nilausen, printed by John Wiley and Sons, 1994, at pgs 11–83.

FIG. **3** is a schematic block diagram of the software architecture of a prior art APPN node **300**. As noted, application **302** executing on an APPN end node, such as EN **202** of network **200**, communicates with another end node, such as EN **212**, through a LU-LU session; LU **304** within each end node functions as both a logical port for the application to the network and as an end point of the communication session. The session generally passes through a path control module **312** and a data link control (DLC) module **316** of the node, the latter of which connects to various network transmission media.

When functioning as an APPN router node, such as NN **206**, an intermediate session routing (ISR) module **305** maintains a portion of the session in each "direction" with respect to an adjacent network node, such as NN **216** of network **200**. In response to receiving the BIND message during session establishment, path control **312** and ISR **305** are invoked to allocate resources for the session. In particular, each NN **206, 216** allocates a local form session identifier (LFSID) for each direction of the session; the LFSID is thereafter appended to the packets in a SNA transmission header (TH) to identify the session context. Collectively, each of these individually-established "local" sessions form the logical communication session between the LUs **304** of the end nodes **202, 212**.

When initiating a session, the application **302** specifies a mode name that is carried within the BIND message and distributed to all APPN network nodes; the LU **304** in each node uses the mode name to indicate the set of required characteristics for the session being established. Specifically, the mode name is used by control point (CP) module **308** of each APPN node **300** to find a corresponding class of service (COS) as defined in a COS table **310**. The CP coordinates performance of all APPN functions within the node, including management of the COS table **310**. The COS definition in table **310** includes a priority level specified by transmission priority (TP) information **320** for the packets transferred over the session; as a result, each APPN network node is apprised of the priority associated with the packets of a LU-LU session. The SNA architecture specifies four (4) TP levels: network priority, high priority, medium priority and low priority. Path control **312** maintains a plurality of queues **314**, one for each TP level, for transmitting packets onto the transmission media via DLC **316**.

Data link switching (DLSw) is a forwarding mechanism for the SNA architecture over an IP backbone network, such as the Internet. A heterogeneous DLSw network is formed when two DLSw switches interconnect the end nodes of the APPN network by way of the IP network; the DLSw switches preferably communicate using a switch-to-switch protocol (SSP) that provides packet "bridging" operations at

the LLC (i.e., DLC) protocol layer. FIG. **4** is a schematic block diagram of a prior art DLSw network **400** comprising DLSw switches **406**, **416** interconnecting ENs **402**, **412** via IP network **410**. The DLSw forwarding mechanism is also well-known and described in detail in *Request for Comment* (RFC) 1795 by Wells & Bartky, 1995 at pgs 1–91.

According to the DLSw technique, a lower-layer DLC connection is established between each EN and DLSw switch; however, these connections terminate at the switches **406**, **416**. In order to provide a complete end-to-end connection between the end nodes, the DLC connections are "disposed" over a reliable, higher-layer transport mechanism, such as TCP sessions. DLSw switches can establish multiple, parallel TCP sessions using well-known port numbers. All packets associated with a particular DLC connection typically follow a single, designated TCP session. Accordingly, SNA data frames originating at a sending EN **402** are transmitted over a particular DLC connection along TR **404** to DLSw switch **406**, where they are encapsulated within a designated TCP session as packets and transported over IP network **410**. The packets are received by DLSw switch **416**, decapsulated to their original frames and transmitted over a corresponding DLC connection of TR **414** to EN **412** in the order received by switch **406** from EN **402**.

Typically, all packets transmitted by DLSw switch **406** over a DLC connection/TCP session flow at the same priority level from a single output queue **405** of the switch and arrive at an output queue **415** of DLSw switch **416** in the same order in which they are transmitted. When the switches are configured as bridges to forward packets over a TCP session through the IP network, prioritization is straightforward. However, it may be desired to integrate the functions of an APPN network node within switch **406** by overlaying an APPN layer onto a DLSw layer of the switch; the resulting hybrid node may prioritize the packets at the APPN layer in an order governed by the TP information levels.

A problem that arises when deploying a hybrid node in such a heterogeneous network is that the TP priority information is lost when passing the packets between the APPN and DLSw layers, primarily because the TP information is not encapsulated within the packets. That is, the APPN layer has knowledge of the TP levels associated with the packets of a LU-LU session as a result of the BIND message exchange during session establishment; yet that information is not encapsulated within the associated packets and, thus, is not conveyed beyond the APPN layer. An example of a tagging mechanism suitable for use with the present invention that conveys TP levels from the APPN layer to the DLSw layer is disclosed in copending and commonly-assigned U.S. patent application, titled *Technique for Maintaining Prioritization of Data Transferred Among Heterogeneous Nodes of a Computer Network*, filed herewith and incorporated by reference as though fully set forth herein.

As described in the commonly-assigned application, the APPN protocol layer of the hybrid node assigns a TP level to each packet and passes that priority information to the DLSw layer of the node via an application programming interface extension. The TP level is converted to information that is "tagged" to each packet and the DLSw layer allocates each tagged packet to a TCP session based on the assigned TP level. The tagged information is then encapsulated within an IP header to enable intermediate routers to maintain the order and priority of the packet as it is transmitted outbound over the IP network to a receiving DLSw switch.

However, the tagged information within the IP header is not discernible to the receiving DLSw switch and, thus, the switch has no knowledge of the TP level associated with the outbound packet. If that packet requests a response, the DLSw switch cannot select, on the basis of priority, the proper TCP session over which to transmit a corresponding inbound packet; accordingly, the switch arbitrarily chooses a session. If the chosen TCP session has a lower designated priority than the session carrying the outbound packet, network throughput may be negatively impacted.

One solution to this problem is to deploy another hybrid node in place of the receiving DLSw switch. This approach is undesirable primarily because a goal of heterogeneous network design is to minimize the number of hybrid nodes in the network. A reason for minimizing the number of hybrid nodes is that such nodes require additional processing and memory resources, thereby resulting in expensive deployments. The present invention is directed to solving the problem of distributing packet prioritization information, assigned by a hybrid node of a heterogeneous network, to switching nodes of the network.

## SUMMARY OF THE INVENTION

The invention comprises a mechanism for conveying information pertaining to transmission priority (TP) levels of inbound packets transmitted over a heterogeneous network from a switching node to a hybrid node of the network. The mechanism comprises a packet-recognizing filter having a novel format that is generated by the hybrid node and dynamically transmitted to the switching node over a predefined communication channel of the network. As described further herein, the filter enables the switching node to classify the inbound packets and assign them appropriate TP levels.

In the illustrative embodiment, the heterogeneous network is preferably a data link switching (DLSw) network with end nodes interconnected by way of an Internet protocol (IP) backbone network and the hybrid node is an advanced peer-to-peer networking (APPN) node with DLSw capabilities. Applications executing on the end nodes communicate via logical unit to logical unit (LU-LU) sessions, whereas the switching node communicates with the APPN node using a switch-to-switch protocol (SSP) over data link control (DLC) connections associated with the LU-LU sessions of the DLSw network; these DLC connections are further overlayed onto existing transmission control protocol (TCP) sessions of the IP network. Preferably, each TCP session is further associated with a TP level.

According to aspects of the invention, the predefined communication channel may be implemented as either an in-band channel over one of the existing TCP sessions using novel extensions to SSP, or an out-band channel over a newly-created TCP session. The format of the filter is preferably customized for each channel implementation; nevertheless, each filter includes a unique opcode identifying the filter, a format identifier (FID) denoting the format of a specific inbound packet, a local form session identifier (LFSID) that classifies the LU-LU session context of the specific packet and a priority identifier specifying the TP level of the packet.

Operationally, an APPN protocol layer of the APPN node passes the op-code, LFSID, FID and priority identifier to a DLSw protocol layer of the node, through an application programming interface (API), during establishment of the LU-LU session. In response to the API, the DLSw layer encapsulates these identifiers within fields of the filter and transfers the filter over the communication channel to the switching node. When transferring the filter over the in-band

communication channel, the opcode is encapsulated within a SSP header, whereas for the out-band channel embodiment, additional addressing information is encapsulated with the opcode in fields of a defined header.

Upon receiving the filter, a DLSw layer of the switching node stores the LFSID, FID and priority identifier and proceeds to examine each inbound packet prior to forwarding it to the APPN node. Specifically, the switching node initially determines the format of each packet and if it matches the stored FID, the node compares the LFSID of the inbound packet with the stored LFSID to identify the LU-LU session context of the packet. If the values of these latter identifiers match, the switching node assigns to the inbound packet the TP level specified by the stored priority identifier and forwards the packet to the APPN node over an appropriate one of the existing TCP sessions.

## BRIEF DESCRIPTION OF THE DRAWINGS

The above and further advantages of the invention may be better understood by referring to the following description in conjunction with the accompanying drawings in which like reference numbers indicate identical or functionally similar elements:

FIG. 1 is a schematic block diagram of prior art communications architecture protocol stacks, such as the Internet protocol stack, used to transmit data between stations of a computer network;

FIG. 2 is a schematic block diagram of a prior art Advanced Peer to Peer Networking (APPN) network including APPN nodes;

FIG. 3 is a schematic block diagram of the software architecture a prior art APPN node;

FIG. 4 is a schematic block diagram of a prior art data link switching (DLSw) network;

FIG. 5 is a block diagram of a heterogeneous computer network, including a DLSw node and an APPN/DLSw hybrid node for interconnecting various subnetworks and communication links on which the present invention may advantageously operate;

FIG. 6 is a schematic block diagram of protocol stacks contained within the DLSw and APPN/DLSw nodes of FIG. 5;

FIG. 7 is a schematic block diagram illustrating the assignment of priority levels among established communication sessions and the distribution of packets among the sessions;

FIG. 8 is a schematic block diagram of a novel packet-recognizing filter generated by the hybrid node of FIG. 5 and dynamically transmitted to the DLSw node over a predefined communication channel in accordance with the invention;

FIGS. 9A and 9B are schematic block diagrams depicting formats of the novel packet-recognizing filter of FIG. 8;

FIG. 10 is a flowchart illustrating use of the novel filter in accordance with the present invention; and

FIG. 11 is a schematic block diagram depicting the format of a conventional transmission header upon which the inventive packet-recognizing filter may advantageously operate.

## DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENT

FIG. 5 is a block diagram of a computer network **500** comprising a collection of interconnected communication links and subnetworks attached to a plurality of stations. The stations are typically computers comprising end stations **502, 512** and intermediate stations **600, 650**. Preferably, the end stations are Advanced Peer to Peer Networking (APPN) end nodes, although the stations may comprise other types of nodes such as Low Entry Networking nodes or Physical Units 2.0 via Dependent Logical Unit Requestor functions. In addition, the intermediate station **650** is a data link switching (DLSw) node and intermediate station **600** is an APPN/DLSw hybrid node.

Each node typically comprises a plurality of interconnected elements, such as a processor, a memory and a network adapter. The memory may comprise storage locations addressable by the processor and adapter for storing software programs and data structures associated with the inventive filtering mechanism and techniques. The processor may comprise processing elements or logic for executing the software programs and manipulating the data structures. An operating system, portions of which are typically resident in memory and executed by the processor, functionally organizes the node by, inter alia, invoking network operations in support of software processes executing on the node. It will be apparent to those skilled in the art that other processor and memory means, including various computer readable media, may be used for storing and executing program instructions pertaining to the techniques described herein.

The subnetworks included within network **500** are preferably local area networks (LANs) and the communication links may include wide area network (WAN) links; in the illustrative embodiment of the invention, the LANs are preferably token rings (TR) **504, 514** and an IP network **510**, which may comprise either a LAN and/or a WAN configuration such as X.25, interconnects the nodes **600, 650**. Communication among the nodes coupled to the network **500** is typically effected by exchanging discrete data packets or frames via connection-oriented service sessions between the communicating nodes.

Heterogeneous (DLSw) network **500** is formed when APPN/DLSw hybrid node **600** is connected to DLSw node **650** via IP network **510**. FIG. 6 is a schematic block diagram of protocol stacks **610, 660** within the nodes **600** and **650**, respectively. Applications executing on SNA devices (end stations) **602, 608** typically access the network through logical units (LUs) of the stations and communicate via LU-LU sessions. Hybrid node **600** functions to facilitate establishment and routing of these connection-oriented communication sessions within the network. To this end, protocol stack **610** preferably comprises an APPN layer **612** that contains the software modules described in FIG. 3.

The stack **610** also includes a Transmission Control Protocol/Internet protocol (TCP/IP) layer **616** containing those layers of the Internet communications architecture protocol stack (FIG. 1) needed to establish, e.g., conventional connection-oriented, TCP communication sessions. Physical sublayers **622** and **626** specify the electrical, mechanical, procedural and functional specifications for activating, maintaining and de-activating the physical links **604** and **605** of the network. Protocol stack **660** of DLSw node **650** likewise includes a TCP/IP layer **666** and physical sublayers **672** and **676**, which are functionally equivalent to those layers of protocol stack **610**.

Each node **600, 650** further contains a DLSw layer **614, 664** and data link control (DLC) layers **618, 620** and **668, 670**, respectively, the latter layers providing a connection-oriented service via conventional DLC connections. The DLSw layers provide a mechanism for forwarding data

frame traffic between devices **602**, **608** over IP network **605**. Preferably, the DLSw layers **614**, **664** cooperate in a peer-relationship and communicate via a switch-to-switch protocol (SSP) to, inter alia, define TCP sessions over the IP network.

In the illustrative embodiment, there are a plurality of connection/session "views" established within the network. For example, from an APPN view, there is a DLC connection **646** between device **602** and APPN layer **612** of node **600**, and a DLC connection **648** between APPN layer **612** and device **608**. From a DLSw view, there is a DLC connection **642** between APPN layer **612** and DLSw layer **614** of node **600**, and a DLC connection **644** between DLSw layer **664** and device **608**; in order to provide reliable, end-to-end connections between the devices, these DLC connections are "overlayed" onto TCP sessions (denoted **645**) between the two DLSw layers **614**, **664**. Lastly, from a LU view, there are multiple LU-LU sessions **680** (at various priority levels) between the LUs of devices **602** and **608**.

It should be noted that the TCP sessions are initiated between DLSw peers **614**, **664** in accordance with a conventional TCP transport protocol. Thereafter, SSP control messages are exchanged between the DLSw layers **614**, **664** of the nodes to establish an end-to-end DLSw circuit over the session. Information contained within these control messages are used to generate a DLSw circuit identifier (ID) that associates the DLSw circuit with the session. Preferably, the DLC connections **642**, **644** overlayed on the TCP session **645** "map" to the DLSw circuit. The generation of DLSw circuits and identifiers is described in *Request for Comment* (*RFC*) 1795 by Wells & Bartky, 1995, while the establishment of multiple TCP sessions between DLSw peer layers is described in both RFC 1795 and *Internetworking with TCP/IP* by Comet and Stevens, printed by Prentice Hall, 1991; all of these publications are hereby incorporated by reference as though fully set forth herein.

Typically, packets transmitted by a DLSw switch over a TCP session flow at the same priority level from a single output queue of the switch and arrive at a peer DLSw switch in the same order in which they are transmitted. Hybrid node **600** may, however, prioritize the packets of a LU-LU session at the APPN layer **612** in an order specified by transmission priority (TP) information contained within the node **600**. FIG. **7** is a schematic block diagram illustrating the assignment of TP levels among established communication sessions and the distribution of packets among those sessions.

A path control module **312** (FIG. **3**) of the APPN layer **612** within node **600** maintains four queues **702-708**, one for each TP level, for transmitting data packets (received from DLC connection **646**) over established TCP sessions **645** of the network. As described above, TCP sessions are established through the IP network **605** in accordance with conventional TCP/IP transport mechanisms within the APPN/DLSw node **600** and the DLSw node **650**; illustratively, these nodes cooperate in a peer-relationship to establish multiple, parallel TCP sessions **732–738** over the network.

The tagging mechanism of the commonly-assigned application incorporated by reference herein allows the hybrid node **600** to convey a TP level from its APPN layer **612** to its DLSw layer **614**, convert that TP level to information that is "tagged" to each outbound packet, and allocate the tagged packet to a TCP session based on the assigned TP level. Specifically, the DLSw layer **614** loads the packets into the queue **725** and then distributes them among four queues **712–718**. Each TCP session (and queue) is preferably associated with a TP level; for example, session **732** (and queue **712**) are assigned a high-priority level, session **734** (and queue **714**) are assigned a medium-priority level, session **736** (and queue **716**) are assigned a normal-priority level and session **738** (and queue **718**) are assigned a low-priority level.

The tagged information is encapsulated within an IP header of the packet prior to outbound transmission over the IP network **605** to the DLSw node **650**. As noted, the tagged information is not discernible to the DLSw node **650** and, if required to respond to the packet, that node cannot select, on the basis of priority, the proper TCP session over which to transmit a corresponding inbound packet because it has no knowledge of the TP level associated with the outbound packet.

In accordance with the present invention, a mechanism is provided for conveying the TP level of an inbound packet transmitted over a heterogeneous network from DLSw node **650** to hybrid node **600**. Referring to FIG. **8**, the mechanism comprises a packet-recognizing filter **800** having a novel format that is generated by the hybrid node **600** and dynamically transmitted to DLSw node **650** over a predefined communication channel **850** of IP network **605**. As described further herein, the filter **800** enables the switching node **650** to classify each inbound packet **810** and assign it an appropriate TP level.

According to an aspect of the invention, the predefined communication channel **800** may be implemented as either an in-band channel over one of the existing TCP sessions **732–738** using novel extensions to SSP, or an out-band channel over a newly-created TCP session. For this latter channel implementation, the newly-created TCP session is established in accordance with the conventional TCP transport protocol described above.

In another aspect of the invention, the format of filter **800** is preferably customized for each channel implementation, as depicted in FIGS. **9A** and **9B**. For each case, the filter includes a well-defined, unique opcode identifying the filter used in the illustrative network configuration, a format identifier (FID) denoting the format of a specific inbound packet, a local form session identifier (LFSID) that classifies the LU-LU session context of the specific packet and a priority identifier specifying the TP level of the packet.

FIG. **9A** illustrates the format **900** of the filter **800** configured for transfer over the in-band communication channel. Here, the opcode **904** is encapsulated by DLSw layer **614** within a SSP header **910** along with the DLSw circuit ID **902**; since the circuit ID **902** associates an end-to-end DLSw circuit with the LU-LU session of the specific inbound packet, additional addressing information is not needed. Such additional addressing information comprises media access control (MAC) addresses of the source node (SMAC) and destination node (DMAC) which, for this example, are nodes **600** and **650**, respectively, and service access points (SAP) addresses of the source (SSAP) and destination (DSAP) nodes.

The format **900** further stores the FID, LFSID and priority identifiers within fields **912–916**, respectively. The contents of these identifiers (along with the additional addressing information) are provided by the APPN layer **612** (FIG.**8**) to the DLSw layer **614** via an application programming interface (API) layer **820** using a data control flow mechanism, such as an API call **825**.

FIG. **9B** illustrates the format **950** of the filter **800** configured for transfer over the out-of-band channel

embodiment. Since a new TCP session is created for this channel embodiment, the additional addressing information passed from the APPN layer **612** to the DLSw layer **614** via the API call **825** is used in format **950**. Specifically, a defined header **970** is generated by DLSw layer **614** for encapsulating the opcode in field **954**, SMAC in field **956**, DMAC in field **958**, SSAP in field **960** and DSAP in field **962**; a value specifying the length of the filter is stored in field **952** of the header **970**. The format **950** further accomodates the FID, LFSID and priority identifiers within fields **972–976**, respectively.

Operation of the present inventive filter mechanism will now be described in connection with the flowchart of FIG. **10**. The operation starts at Step **1000** and proceeds to Step **1010** where APPN protocol layer **612** of hybrid node **600** passes filter-identifying information, such as the opcode, LFSID, FID and priority identifier, to DLSw protocol layer **614** through API layer **820** during establishment of the LU-LU session. In response to the API, the DLSw layer encapsulates these identifiers within fields of the filter as described above and transfers the filter over the communication channel **850** to the DLSw switching node **650** in Step **1015**.

Upon receiving the filter, DLSw layer **664** of the switching node **650** interprets the opcode as describing the type of message as a filter, parses the fields of the filter and stores the LFSID, FID and priority identifier in a temporary storage location **805** of, e.g., the memory of node **650** (Step **1020**). The layer **664** then proceeds to examine each inbound packet **810** prior to forwarding it to the APPN node **600**. The inbound packets are typically Systems Network Architecture (SNA) type frames generated by SNA devices coupled to the DLSw network; these frames are, in turn, typically encapsulated with conventional SNA transmission header (TH) headers. FIG. **11** is a schematic block diagram of the format **1100** of the TH header that DLSw layer **664** of node **650** is configured to operate on using the packet-recognizing filter **800**.

Specifically, the switching node determines the format of each inbound packet by initially examining the contents of a FID type field **1110** of the TH header and comparing them with the stored FID identifier in Step **1025**. It should be noted that the switching node is configured to recognize the format of TH header and, thus, can access the contents of any particular fields contained therein. The SNA architecture defines several different types of packet formats; in the illustrative embodiment, a FID2 format is preferably used for communication among the SNA devices **602,608**. If the contents of the FID type field do not match the contents of the stored FID (Step **1030**), a default priority is assigned to the inbound packet by the switching node (Step **1035**).

If there is a match in Step **1030**, the node then accesses the contents of the LFSID field **1120** of the header and compares those contents with the contents of the stored LFSID to identify the LU-LU session context of the packet (Step **1040**). If the values of the LFSIDs do not match (Step **1045**), a default priority is assigned in Step **1046**; otherwise, the switching node assigns to the inbound packet the TP level specified by the contents of the stored priority identifier in Step **1050**. That is, the DLSw layer **664** maps the packet to a selected TCP session **732–738** (FIG. **7**) based on the TP level specified by the priority identifier and loads the packet onto a corresponding queue **772–778** associated with the selected session. TCP/IP driver code within layer **666** of node **650** (FIG. **6**) then maps the TP designation of the packet to a predetermined value of precedence bits and configures those bits as a "tag" within a type of is service (TOS) field when building an IP header for the packets during TCP session establishment. Thereafter, the prioritized packet is transferred through the core IP backbone network **605** to the hybrid node **600** in Step **1055** and the operation ends in Step **1060**.

Advantageously, the present invention enables the APPN layer of a hybrid node to instruct the DLSw layer of a receiving switching node as to the TP level of an inbound packet destined for the hybrid node. In response to the instruction, the switching node assigns the inbound packet to a particular TCP session, where priority is preserved at intermediate queuing points on the basis of the value of the precedence bits in the IP header.

While there has been shown and described an illustrative embodiment for conveying information pertaining to priority levels of inbound packets transmitted over a heterogeneous network from a switching node to a hybrid node of the network, it is to be understood that various other adaptations and modifications may be made within the spirit and scope of the invention. For example in an alternate embodiment of the invention, a different transport mechanism may be employed in the heterogeneous network to transport different packet formats among the end nodes, such as NetBIOS devices, of the network. Here, another unique, yet well-defined opcode is needed to identify the filter used to convey the priority levels of these different inbound packets over the alternately-configured network.

The foregoing description has been directed to specific embodiments of this invention. It will be apparent, however, that other variations and modifications may be made to the described embodiments, with the attainment of some or all of their advantages. Therefore, it is the object of the appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.

What is claimed is:

**1**. Apparatus for conveying information pertaining to transmission priority (TP) levels of inbound packets transmitted over a heterogeneous network from a switching node to a hybrid node of the network, the apparatus comprising:

a predefined communication channel interconnecting the hybrid and switching nodes; and

a packet-recognizing filter generated by the hybrid node and dynamically transmitted to the switching node over the predefined communication channel, the filter enabling the switching node to classify the inbound packets and assign them appropriate TP levels.

**2**. The apparatus of claim **1** wherein the predefined communication channel is one of an in-band channel over an existing transport session connection between the nodes and an out-of-band channel over a newly-created transport session between the nodes.

**3**. The apparatus of claim **2** wherein the filter comprises identifiers identifying attributes of the inbound packets, such that inbound packets matching these identifiers are associated with appropriate TP levels.

**4**. The apparatus of claim **3** wherein the identifiers comprise a local form session identifier classifying the session context of a specific inbound packet and a priority identifier specifying the TP level of the packet.

**5**. The apparatus of claim **4** wherein the hybrid node comprises an Advanced Peer to Peer Networking (APPN) protocol layer and a Data Link Switching (DLSw) protocol layer, and wherein the APPN protocol layer passes the identifiers to the DLSw protocol layer of the hybrid node through an application programming interface.

6. The apparatus of claim 5 wherein the DLSw protocol layer encapsulates the identifiers within fields of the filter and transfers the filter over the predefined communication channel.

7. The apparatus of claim 6 wherein the filter further comprises a unique opcode identifying the filter, and wherein the opcode is encapsulated within a switch-to-switch protocol header when transferring the filter over the in-band channel.

8. The apparatus of claim 6 wherein the filter further comprises a unique opcode identifying the filter, and wherein the opcode and additional addressing information are encapsulated within a defined header when transferring the filter over the out-of-band channel.

9. A method for conveying information pertaining to transmission priority (TP) levels of inbound packets transmitted over a heterogeneous network from a switching node to a hybrid node of the network, the method comprising the steps of:

establishing a predefined communication channel interconnecting the hybrid and switching nodes;

generating a packet-recognizing filter at the hybrid node;

dynamically transmitting the filter to the switching node over the predefined communication channel; and

assigning the inbound packets TP levels at the switching node using the filter.

10. The method of claim 9 wherein the hybrid node comprises an Advanced Peer to Peer Networking (APPN) protocol layer and a Data Link Switching (DLSw) protocol layer, and wherein the filter comprises identifiers such as a unique opcode identifying the filter, a format identifier (FID) denoting the format of a specific inbound packet, a local form session identifier (LFSID) classifying the session context of the specific inbound packet and a priority identifier specifying a TP level of the packet.

11. The method of claim 10 wherein the step of generating comprises the step of passing the identifiers from the APPN protocol layer to the DLSw protocol layer of the hybrid node through an application programming interface (API).

12. The method of claim 11 wherein the step of generating further comprises the step of encapsulating the identifiers within fields of the filter at the DLSw layer in response to the API.

13. The method of claim 12 wherein the predefined communication channel is an in-band channel over one of a plurality of existing transport session connections between the nodes or an out-band channel over a newly-created transport session between the nodes.

14. The method of claim 13 wherein the step of generating further comprises one of the steps of:

when transferring the filter over the one in-band channel, encapsulating the opcode within a switch-to-switch protocol header; and

when transferring the filter over the out-band channel, encapsulating the opcode and additional addressing information within fields of a defined header.

15. The method of claim 14 further comprising, after the step of dynamically transferring, the steps of:

receiving the filter at a DLSw protocol layer of the switching node;

parsing the fields of the filter;

storing the LFSD, FID and priority identifier in a temporary storage location of the switching node; and

examining each inbound packet prior to forwarding the inbound packet to the hybrid node.

16. The method of claim 15 wherein the step of examining further comprises the steps of:

initially determining the format of the inbound packet;

if the format of the inbound packet is equal to the format specified by the stored FID, comparing the LFSID of the inbound packet with the stored LFSID; and

if the LFSIDs match, assigning the inbound packet the TP level specified by the stored priority identifier.

17. The method of claim 16 further comprising the step of, after the step of assigning, forwarding the inbound packet to the hybrid node over an appropriate one of the existing transport session connections.

18. A computer readable medium containing executable program instructions for conveying information pertaining to transmission priority (TP) levels of inbound packets transmitted over a heterogeneous network from a switching node to a hybrid node of the network, the executable program instructions comprising program instructions for:

generating a packet-recognizing filter at the hybrid node;

dynamically transmitting the filter to the switching node over a predefined communication channel interconnecting the hybrid and switching nodes, wherein the predefined communication channel is an in-band channel over one of an existing transport session connection between the nodes and an out-band channel over a newly-created transport session between the nodes; and

assigning the inbound packets TP levels at the switching node using the filter.

19. The medium of claim 18 wherein the program instructions for generating comprises program instructions for encapsulating identifiers within fields of the filter.

20. The medium of claim 19 wherein the program instructions for generating further comprises program instructions for:

encapsulating the opcode within a first header when transferring the filter over the in-band channel; and

encapsulating the opcode and additional addressing information within fields of a second header when transferring the filter over the out-band channel.

21. Apparatus for conveying information pertaining to transmission priority (TP) levels of inbound packets transmitted over a heterogeneous network from a switching node, the apparatus comprising:

a hybrid node for being coupled to a predefined communication channel for interconnecting the hybrid node and the switching node, the hybrid node being configured to generate a packet-recognizing filter for being transmitted to the switching node over the predefined communication channel, the filter enabling the switching node to classify the inbound packets and assign them appropriate TP levels.

* * * * *

US005473607A

# United States Patent [19]

Hausman et al.

[11] Patent Number: 5,473,607

[45] Date of Patent: Dec. 5, 1995

[54] **PACKET FILTERING FOR DATA NETWORKS**

[75] Inventors: **Richard J. Hausman**, Soquel; **Lazar Birenbaum**, Saratoga, both of Calif.

[73] Assignee: **Grand Junction Networks, Inc.**, Union City, Calif.

[21] Appl. No.: **103,659**

[22] Filed: **Aug. 9, 1993**

[51] **Int. Cl.**[6] ..... **H04L 12/28**

[52] **U.S. Cl.** ..... **370/85.13**; 370/94.1; 370/92

[58] **Field of Search** ..... 370/60, 60.1, 85.13, 370/85.14, 94.1, 94.2, 94.3, 92, 85.3; 395/159, 118, 400

[56] **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 4,399,531 | 8/1983 | Grande et al. | 370/60 |
| 4,627,052 | 12/1986 | Homre et al. | 370/85.13 |
| 4,679,193 | 7/1987 | Jensen et al. | 370/94.1 |
| 4,891,803 | 1/1990 | Juang et al. | 370/60 |
| 4,933,937 | 6/1990 | Konishi | 370/85.13 |
| 5,032,987 | 7/1991 | Broder et al. | 364/200 |
| 5,210,748 | 5/1993 | Onishi et al. | 370/60 |
| 5,218,638 | 6/1993 | Matsumoto et al. | 380/23 |
| 5,247,620 | 9/1993 | Fukuzawa et al. | 370/85.13 |
| 5,274,631 | 12/1993 | Bhardwaj | 370/60 |

OTHER PUBLICATIONS

Kalpana, Etnerswithc Product Overview, Mar. 1990, pp. 1–20.

Artel, Galactica Stenbridg C/802.3 Application Note, Nov. 1991, pp. 1–26.

Synernetics, Lanplex 5000: Intra–Network Banowidth, 1992, pp. 1–12.

Synernetics, Lanplex 5000 Intelligent Switching Hubs, 1993, pp. 1–9.

Synernetics, Lanplex 5000 Family, 1992. pp. 1–6.

Synernetics, Etheract Express Module, 1991, pp. 1–4.

Alantec, Powerhub Arthitectures, Dec. 1992, pp. 1–6.

Bradner, Scott O., "Ethernet Bridges and Routes", Feb. 1992, pp. 1–10, Data Communications.

Kwok, Conrad K. and Biswanath Mjkherjec, "Cut–Through Bridging for CSMA/CE Local Area Network", Jul. 1990, pp. 938–942, I EEE Transactions on Communications, vol. 38, No. 7.

*Primary Examiner*—Douglas W. Olms
*Assistant Examiner*—Chau T. Nguyen
*Attorney, Agent, or Firm*—Michael J. Hughes

[57] **ABSTRACT**

An improved partial packet filter (**10**) for filtering data packets (**210**) in a computer network (**12**) wherein a candidate field (**413**) of the data packet (**210**) is hashed to a plurality of bit-wise subsets (**636**) each being an independent representation of the candidate field (**413**). Each of the bit-wise subsets (**636**) is compared to a reference hash table (**644**) which has been prepared in a preliminary operation series (**514**). The preliminary operation series (**512**) configures a plurality of target fields (**714**) to set selected memory locations (**312**) in the reference hash table (**644**).

**17 Claims, 4 Drawing Sheets**

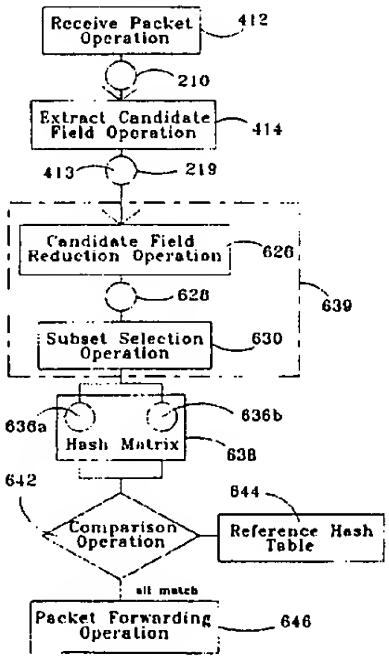


Fig. 1

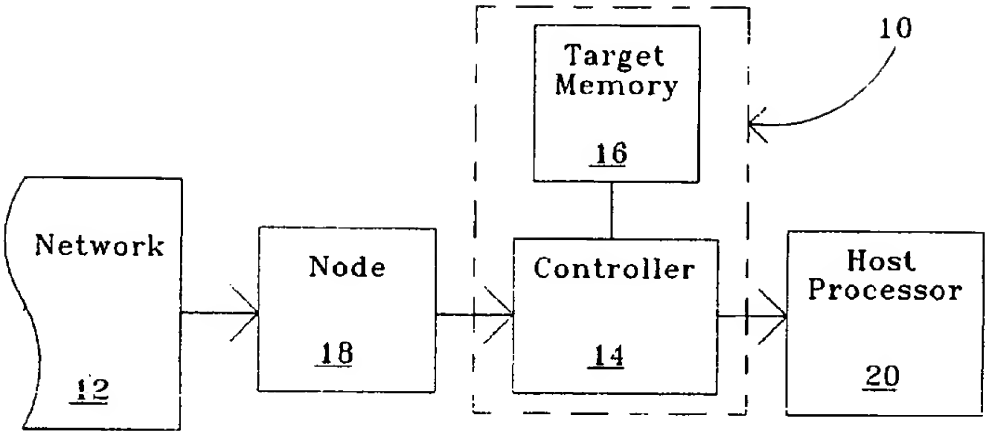

10
Target Memory
16
Network
12
Node
18
Controller
14
Host Processor
20

Fig. 2

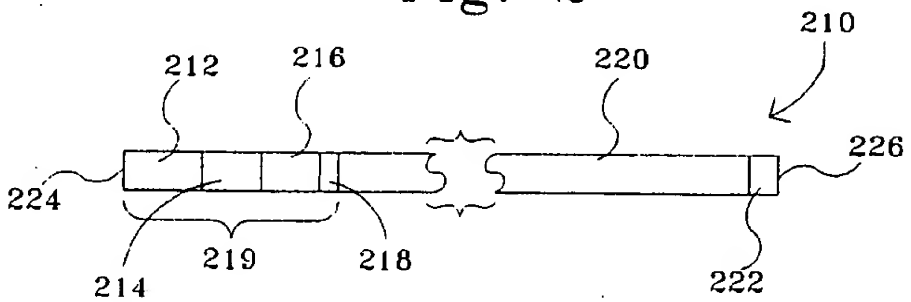

210
212
216
220
226
224
214
219
218
222

Fig. 3

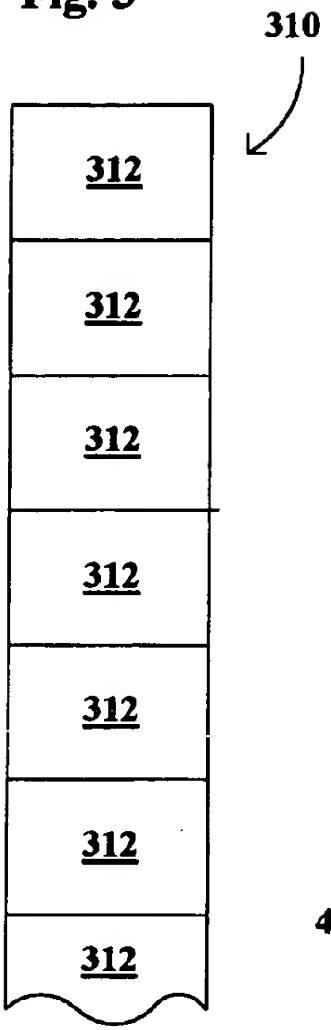



Fig. 4 (prior art)

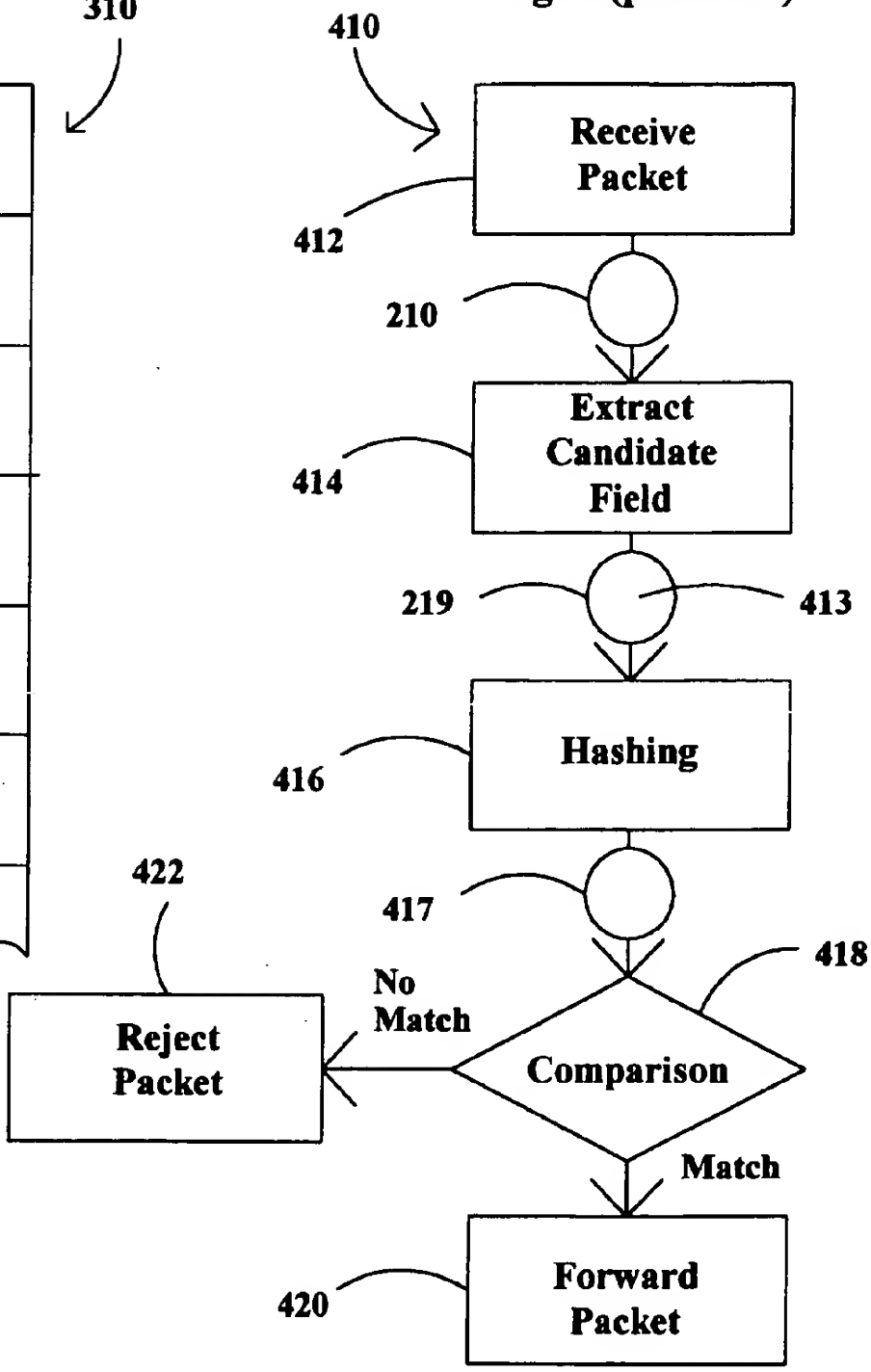


Fig. 5

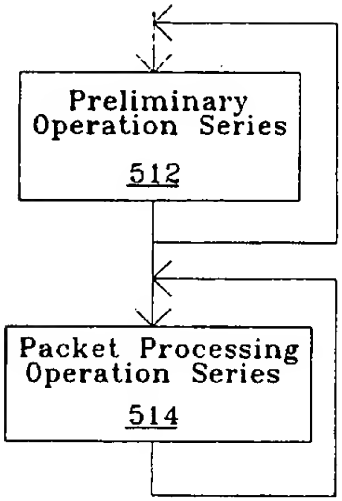

Preliminary Operation Series
512
Packet Processing Operation Series
514

Fig. 6

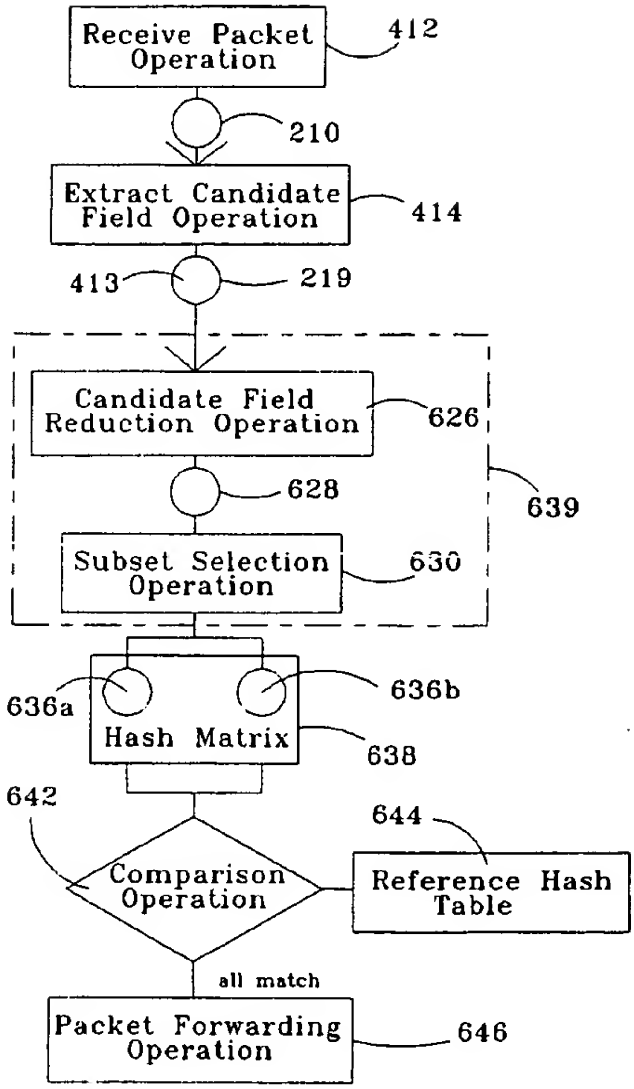
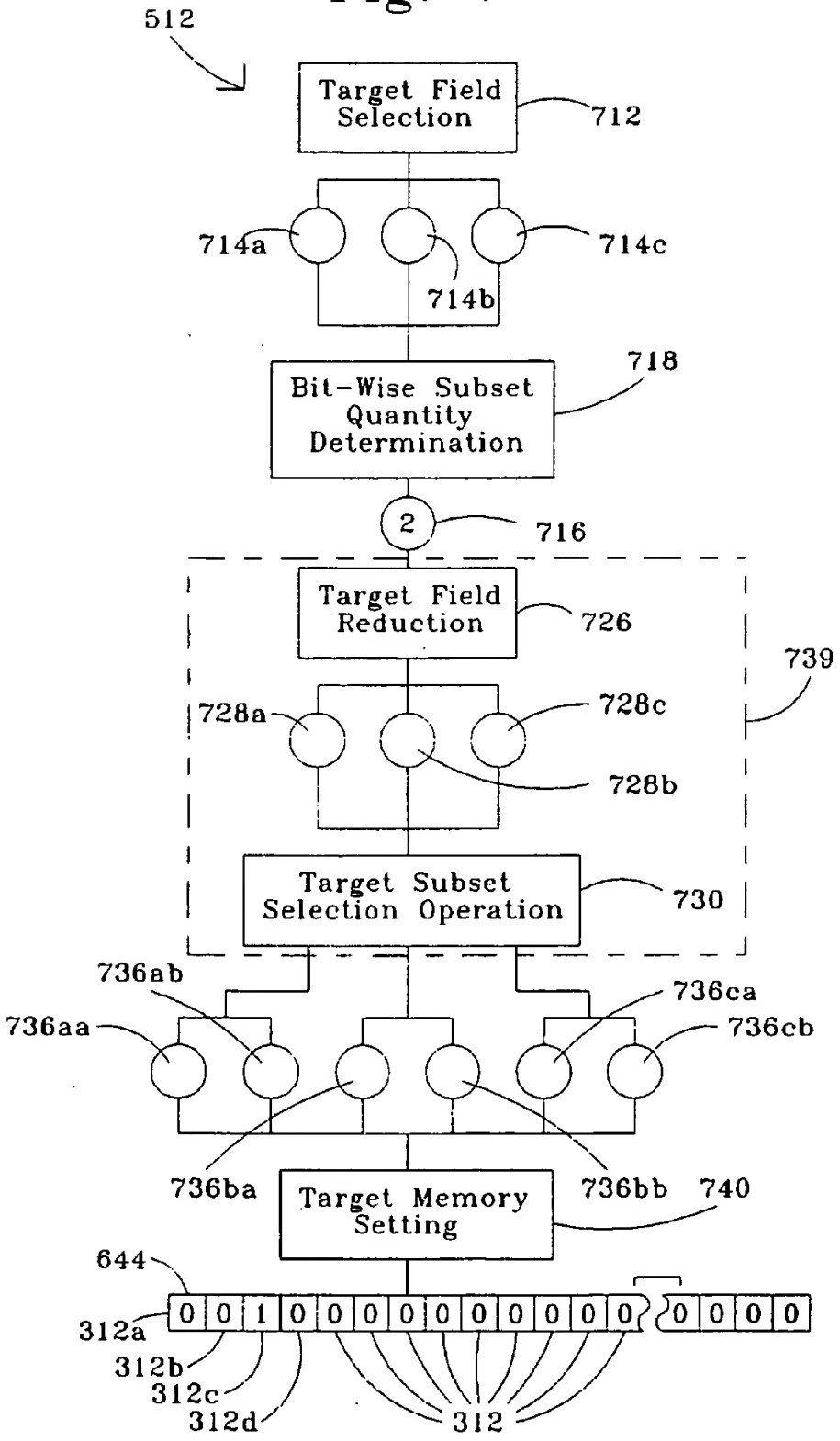

Receive Packet Operation
412
210
Extract Candidate Field Operation
414
413
219
Candidate Field Reduction Operation
626
628
639
Subset Selection Operation
630
636a
636b
Hash Matrix
638
642
644
Comparison Operation
Reference Hash Table
all match
Packet Forwarding Operation
646

Fig. 7


512
Target Field Selection
712
714a
714b
714c
Bit-Wise Subset Quantity Determination
718
2
716
Target Field Reduction
726
739
728a
728b
728c
Target Subset Selection Operation
730
736ab
736aa
736ca
736cb
736ba
736bb
Target Memory Setting
740
644
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
312a
312b
312c
312d
312

# PACKET FILTERING FOR DATA NETWORKS

## TECHNICAL FIELD

The present invention relates generally to the field of computer science and more particularly to data networking and component devices attached to data networks.

## BACKGROUND ART

Computer networks are becoming increasingly common in industry, education and the public sector. The media over which data are carried generally carry data in units referred to as "packets" which are destined for many different sources. Addressing and packet typing are included in most standardized and proprietary packet based networking protocols which make use of destination address fields at the beginning of and/or within each data packet for the purpose of distinguishing proper recipient(s) of the data of the packets. As a packet is received at intermediate and end components in a system, rapid determination of the proper recipient (s) for the data must be made in order to efficiently accept, forward, or discard the data packet. Such determinations are made based upon the above discussed address, packet type and/or other fields within the relevant packets. These determinations can be made by network controller hardware alone, by a combination of hardware and software, or by software alone. In broadcast type networks, every node is responsible for examining every packet and accepting those "of interest", while rejecting all others. This is called "packet filtering". Accuracy, speed and economy of the filtering mechanism are all of importance.

When the above discussed determinations are made through a combination of hardware and software, the hardware is said to have accomplished a "partial filtering" of the incoming packet stream. It should be noted that one type of packet filtering is accomplished on the basis of packet error characteristics such as collision fragments known as "runts", frame check sequence errors, and the like. The type of filtering relevant to the present discussion is based upon packet filtering in which filtering criteria can be expressed as simple Boolean functions of data fields within the packet as opposed to filtering based upon detection of errors or improperly formed packets.

In the simplest case, each node of a computer network must capture those packets whose destination address field matches the node's unique address. However there frequently occur situations in which additional packets are also of interest. One example occurs when the node belongs to a predefined set of nodes all of which simultaneously receive certain specific "groupcast" packets which are addressed to that group. Groupcast packets are usually identified by some variation of the address field of the packet. Groupcast address types generally fall into one of two forms. "Broadcast" addresses are intended for all nodes and "multicast" addresses are targeted for specific applications to which subsets of nodes are registered. Another case of such field-based packet filtering occurs when certain network management nodes are adapted to focus on specific protocols, inter-node transactions, or the like, to the exclusion of all other traffic.

Attachment of a networked device to the network is realized through a "controller" which operates independently of the host processor. Packet filtering then occurs in two successive stages beginning at the controller, which examines packets in real-time. To accomplish this, the



controller is "conditioned" with an appropriate subset of the specified filtering criteria, according to the filtering capabilities of that controller. The controller classifies packets into three categories: Those not satisfying the filter criteria ("rejects"); those satisfying the criteria ("exact matches"); and those possibly satisfying the criteria ("partial matches"). Rejects are not delivered to the processor. Those packets which are classified as exact or as possible matches are delivered, with appropriate indications of their classification, to the device processor. The controller, ideally, excludes as many unwanted packets as its capabilities will allow, and the host processor (with the appropriate software operating therein) completes the overall filtering operation, as required. The value of filtering packets at the controller level (the partial filtering) is that it reduces the burden on the host processor.

Controller filtering implementations are constrained by the fact that they must process packets in real-time with packet reception. This places a high value on filtering mechanisms that can be implemented with a minimum amount of logic and memory. Controller based filtering criteria are contained in a target memory. In the case of exact matching, a literal list of desired targets is stored in the target memory. While exact matching provides essentially perfect filtering, it can be used in applications wherein there are only a very small number of targets.

Partial filtering is employed when the potential number of targets is relatively large, such as is often the case in multicast applications. A primary consideration is the "efficiency" of the partial filter. Efficiency (E), in this context, may be expressed as:

$$E=Tn/Pn$$

where:

Tn=the number of target packets of interest; and

Pn=the number of potential candidates delivered to the processor.

An efficiency of E=1.0 represents an exact filtering efficiency wherein every candidate is a desired target. This is the efficiency of the filtering which occurs in the "exact matching" previously discussed herein. While exact filtering efficiency is an objective, the previously mentioned constraints, including that the controller must do its filtering in essentially real-time, will generally not allow for such efficiency.

The predominant method used in the prior art for partial packet filtering is "hashing". The process conventionally begins with the extraction from each received packet of all fields involved in the specified filtering criteria. The composite of such relevant fields is called the "candidate field". Assuming an even distribution of candidate fields (a situation that is not always literally accurate, but the assumption of which is useful for purposes of analysis), there will be a potential number of packet candidates of $2^{Cb}$ where Cb is the number of bits in the candidate field. The hashing function produces a reduction in the bit size of the candidate field according to a "hashing function". As a part of the initiation of the controller, the hashing function is applied to each field of the target memory to assign a "target hash value" to each such field. The controller memory is initialized as a bit mask representing the set of target hash values. Then, during operation, a "candidate hash value" is created by applying the hashing function to each candidate field. The candidate hash value is used as a bit index into the controller memory, with a match indicating a possible candidate.

As can be appreciated in light of the above discussion and

from a general understanding of simple hashing operations, the hashing function has the effect of partitioning the $2^{Cb}$ candidate possibilities into Mb groups (called "buckets"), where Mb is the number of bits in the controller's target memory. Because candidate packets that fall into the same bucket are not distinguished, a "hit" represents any of $2^{Cb}$/Mb candidates. Useful hashing functions will partition the candidate possibilities in a roughly uniform distribution across the set of Mb buckets. For a single target, the efficiency of such a hashing method is $Mb/2^{Cb}$ If Tn desired targets are represented by Bn buckets (where Bn<=Tn and Bn<=Mb, the efficiency of such a hashing method is:

$$E=Tn/(Bn2^{Cb}/Mb)=TnMb/Bn2^{Cb}$$

In exact matching, target memory could hold Mb/Cb targets. Hashing is appropriate when the number of buckets (Bn) is larger than this figure. However, effective hashing also requires that the number of buckets be less than Mb, because as target memory density increases there is less differentiation among candidate fields. With the target memory full of hash targets, Bn=Mb and the efficiency is $Tn/2^{Cb}$.

As can be appreciated, the described prior art hashing method used for partial packet filtering implies a loss of information in that a single hash value potentially represents a large set of candidates. Clearly, it would be desirable to reduce such loss of data. Correspondingly, it would desirable to maximize the filtering efficiency for a given Mb or (or to minimize the Mb for a given filter efficiency).

To the inventor's knowledge, no prior art method for partial packet filtering has improved efficiency or reduced data loss as compared to the conventional hashing method described above.

## DISCLOSURE OF INVENTION

Accordingly, it is an object of the present invention to provide a method and means for efficiently performing a partial filtering operation on data packets in a computer network.

It is another object of the present invention to provide a method and means for partial packet filtering which rejects a maximum number of incoming packets which are not at interest without requiring a large target memory and without unduly slowing down the processing of incoming packets.

It is still another object of the present invention to provide a partial packet filtering method and means which is inexpensive to implement.

It is yet another object of the present invention to provide a partial packet filtering method and means which will operate in real-time or near real-time.

It is still another object of the present invention to provide a partial packet filtering method and means which is adaptable to a variety of network system requirements.

Briefly, the preferred embodiment of the present invention implements multiple independent hashing functions applied in parallel to the candidate field of each packet. The combined application of multiple independent hashing functions results in specification of a hash matrix, with each coordinate of the hash matrix being the result of one of the hashing functions. The hash matrix includes the results of different hashing algorithms applied to a single candidate field, or the same hashing function applied to different subsets of the candidate field, or a combination thereof. The filter parameters consist of the set of acceptable result values for each hashing operation.

An advantage of the present invention is that partial packet filtering efficiency is improved, thereby freeing the host processor from a substantial portion of the packet filtering operation.

Yet another advantage of the present invention is that filtering efficiency is increased geometrically with an increase in target memory.

Still another advantage of the present invention is that a minimum amount of target memory is required for a specific target efficiency.

Yet another advantage of the present invention is that the partial packet filtering can be performed in a minimum amount of time for a given target efficiency.

These and other objects and advantages of the present invention will become clear to those skilled in the art in view of the description of the best presently known modes of carrying out the invention and the industrial applicability of the preferred embodiments as described herein and as illustrated in the several figures of the drawing.

## BRIEF DESCRIPTION OF THE DRAWING

FIG. **1** is a block diagram depicting a portion of a computer network with an improved partial packet filter according to the present invention in place therein;

FIG. **2** is a diagrammatic representation of a conventional prior art Ethernet data packet;

FIG. **3** is diagrammatic representation of a hash table;

FIG. **4** is a flow chart showing a conventional prior art partial packet filtering operation;

FIG. **5** is a block depiction of a partial packet filtering method according to the present invention;

FIG. **6** is a flow chart, similar to the chart of FIG. **4**, depicting the packet processing operation series of FIG. **5**; and

FIG. **7** is a flow chart depicting the preliminary operation series of FIG. **5**.

## BEST MODE FOR CARRYING OUT INVENTION

The best presently known mode for carrying out the invention is a partial packet filter for implementation in a personal computer resident Ethernet controller. The predominant expected usage of the inventive improved packet filter is in the interconnection of computer devices, particularly in network environments where there are relatively few targets.

The improved partial packet filter of the presently preferred embodiment of the present invention is illustrated in a block diagram in FIG. **1** and is designated therein by the reference character **10**. In the diagram of FIG. **1**, the improved partial packet filter **10** is shown configured as part of a network system **12** (only a portion of which is shown in the view of FIG. 1). In many respects, the best presently known embodiment **10** of the present invention is structurally not unlike conventional partial packet filter mechanisms. Like prior art conventional partial packet filters, the best presently known embodiment **10** of the present invention has a controller **14** with an associated target memory **16**. In the example of FIG. **1**, the improved partial packet filter **10** receives data from a network node **18** and performs the inventive improved packet filtering process on such data before passing selected portions of the data on to a host

processor **18** to which the improved partial packet filter **10** is dedicated.

FIG. **2** is a diagrammatic representation of a conventional Ethernet data packet **210**. The standardized Ethernet packet **210** has a preamble **212** which is 64 bits in length, a destination address **214** which is 48 bits in length, a source address **216** which is 48 bits in length, a length/type field **218** which is 16 bits in length and a data field **220** which is variable in length from a minimum of 46 eight bit bytes to a maximum of 1500 bytes. Following the data field **220** in the packet **210** is a 4 byte (32 bit) frame sequence check ("FCS") **222**. The packet **210** is transmitted serially beginning at a "head" **224** and ending at a "tail" **226** thereof. The preamble **212**, destination address **214**, source address **216** and length/type field **218** are collectively referred to as the header **219**.

FIG. **3** is a diagrammatic representation of a conventional single dimensional hash table **310** with which one skilled in the art will be familiar. The hash table **310** has a plurality of address locations **312** each of which can be "set" (set to 1) or left unset (set to zero).

FIG. **4** is a flow diagram depicting the operation of a conventional prior art partial packet filtering operation **410**. As previously discussed briefly, a packet **210** (FIG. **2**) is received (receive packet operation **412**) from the network **18** (FIG. **1**) and a candidate field **413** (such as the header **219** of the packet **210**) is extracted (extract candidate field operation **414**). A hashing operation **416** is performed on the extracted candidate field **413** to produce a hash value **417** and the hash value **417** is compared to the hash table **310** (FIG. **3**) stored in the target memory **16** (FIG. **1**) in a comparison operation **418**. If the result of the comparison operation **418** is a match, the packet **210** is forwarded in a forward packet operation **420**. If the result of the comparison operation **418** is not a match, the packet **210** is rejected **422** in a reject packet operation. It should be remembered that the use of the header **219** here is an example only, and any portion or combined portions of the packet **210** might constitute the candidate field **413** in a given application.

FIG. **5** is a flow diagram depicting the inventive improved packet filtering process **510**. The improved packet filtering process **510** is accomplished in a preliminary operation series **512** and a packet processing operation **514**, each of which is repeated as required, as will be discussed hereinafter. The preliminary operation series **512** is accomplished according to software residing in the host processor **20** (FIG. **1**) to configure the target memory **16** (FIG. **1**) as will be discussed hereinafter. It should be noted that the fact that the improved packet filtering process **510** is divided into the two main operation categories (the preliminary operation series **512** and the packet processing operation **514**) does not distinguish this invention over the prior art. Rather, the processes within the preliminary operation series **512** and the packet processing operation **514** describe the essence of the inventive process.

FIG. **6** is a flow chart showing the inventive packet processing operation **514** in a manner analogous to the presentation of the prior art partial packet filtering operation **410** depicted in FIG. **4**. As can be seen in the view of FIG. **6**, the packet processing operation series **514** is similar in many respects to the prior art partial packet filtering process **410** (FIG. **4**). In the packet processing operation series **514**, a packet **210** (FIG. **2**) is received (receive packet operation **412**) and a candidate field **413** is extracted in an extract candidate field operation **414**. In the best presently known embodiment **10** of the present invention, the inventive packet processing operation series **514** next performs a candidate field reduction operation **626**. In the best presently known embodiment **10** of the present invention, the candidate field reduction operation **626** is merely the application of the conventional CRC polynomial algorithm to the candidate field **413** to yield a 32 bit CRC output value **628** (although any of a number of similar algorithms might be applied for this purpose). Next, a subset selection operation **630** selects a predetermined number (two in the example of FIG. **6**) of bit-wise subsets **636** from the CRC output value **628**. The method for determining the quantity of bit-wise subsets **636** to be selected in the subset selection operation **630**, and the size of each, will be discussed hereinafter. In the best presently known embodiment **10** of the present invention, the bit-wise subsets **636** are each 6 bits in length. It should be noted that, in the best presently known embodiment **10** of the present invention, the bit-wise subsets **636** are selected from the CRC output value **628** simply by taking the first 6 bits of the CRC output value **628**, the second six bits, and so on until as many bit-wise subsets as are needed are obtained and so, in the best presently known embodiment **10** of the present invention, the bit wise subset **636** are "consecutive bit section " of the fixed size field (the CRC output value **628** in the best presently known embodiment **10** of the present invention. The inventors have determined that the bits of the CRC output value **628** (resulting from the CRC polynomial function) are independent of each other, and so any 6 bit portion of the CRC output value **628** is as representative of the CRC output value **628** as is any other 6 bit portion.

The bit-wise subsets **636** are then compared to the hash table **310** (FIG. **3**) stored in the target memory **16** (FIG. **1**) in a comparison operation **642**. The combined multiple hash values **636** may be considered to be a hash matrix **638** (in the example of FIG. **6**, a two dimensional hash matrix **638**).

It is important to note that the essence of the present inventive method lies in the extraction of the plurality of independent or relatively independent representative indices of the candidate field **413** ("candidate filled indices") which, in the example of the best presently known embodiment **10** of the present invention are the bit-wise subsets **636** which make up the hash matrix **638**. That is, the bit-wise subsets **636** are representative fields in that the bit-wise subsets **636** are representative of the candidate field **413**, as discussed above. The generally simultaneous (parallel) processing of these is the source of the advantages of the present inventive method and means. The exact method described herein in relation to the best presently known embodiment **10** of the present invention, that of first reducing the candidate field **413** in the candidate field reduction operation **626** and then extracting the bit-wise subsets **636** is but one of many potential methods for accomplishing such a parallel hashing operation **639**, and the present invention is not intended to be limited by this aspect of the best presently known embodiment **10**.

In the best presently known embodiment **10** of the present invention, in a comparison operation **642**, each of the bit-wise subsets **636** is compared to a reference hash table **644** (a "target hash array") stored in the target memory **16** (FIG. **1**) and only if all match is the packet **210** forwarded in a packet forwarding operation **646**. In the example of FIG. **6**, the reference hash table **644** will be a 64 element array representing all values from 0 through 63 inclusive. Some elements of the reference hash table **644** are set as will be discussed hereinafter in relation to the preliminary operation series **512**. If the value of the bit-wise subset "falls into one of the buckets" (is equivalent to a corresponding set bit in

the reference hash table **644**), then the data packet **210** is defined as being a "match".

Now returning to a consideration of the preliminary operation series **512** (FIG. **5**) with an understanding of the packet processing operation series **514**, the target memory **16** is configured in process steps much like those described in relation to the packet processing operation series **514** of FIG. **6**.

FIG. **7** is a flow diagram of the preliminary operation series **512** according to the best presently known embodiment **10** of the present invention. A preliminary operation which is common to both the prior art and the present invention is a target field(s) selection process **712**. The target (field) s selection process is merely the selection of criteria to which incoming packets **210** are to be compared. For example, if the entire process is to be on the basis of desired destinations, then an intended destination address **214** (FIG. **2**) will be (one of) the target field(s) **714**, and if three destinations are of interest, then there will be three target fields **714** as illustrated in the example of FIG. **7**. The actual process involved in selecting the target field(s) is a function of network control software which is found in the prior art and which is not relevant to the present invention except to the extent that it delivers the target field(s) **714** to the inventive preliminary operation series **512**.

Having determined the quantity of target fields **714** of interest, host software will next determine a bit-wise subset quantity **716** (the appropriate "subset quantity" of bit-wise subset **636**) in a bit-wise subset quantity determination operation **718**. The bit-wise subset quantity determination operation **718** will be discussed in more detail hereinafter, as it can be better understood in light of the present description of the entire preliminary operation series **512**. For the present simplified example of FIGS. **6** and **7**, and as already mentioned, the bit-wise subset quantity **716** is two. That is, two of the bit-wise subsets **636** are to be extracted from the CRC output value **628** in the subset selection operation **630** of FIG. **6**.

As can be appreciated, the target fields **714** are each equivalent in form to the candidate fields **413** discussed previously herein, and processing of the target fields **714** is much the same as has been previously described herein in relation to the candidate fields **413**. In the inventive preliminary operation series **512**, each of the target fields **714** is processed in a target field reduction operation **726** by application of the CRC polynomial to produce a target CRC value **728**. Each of the target CRC values **728** is then processed in a target subset selection operation **730** to produce a plurality (two for each target CRC value **728** for a total of six, in the present example) of target bit-wise subsets **736**. In more general terms, each of the "target fields **714** (having been selected according to prior art methods as discussed previously, herein) is processed as described to produce a "target representative field" (the target CRC value **728** in the present example), which is then further processed as described to produce the "target indices", which target indices may be "target string subsets**38** of the target representative field and which are, in the present example, the target bit-wise subsets **736**. This process is alike to the process which is repeated as necessary to process each incoming data packet **210**, wherein the candidate fields **413** are processed to produce a candidate representative field (the CRC output value **628** in the present example), which is further processed to produce the "candidate string subsets" (the bit-wise subsets **636** in the present example). The quantity of target bit-wise subsets **736** taken from each target CRC value **728** is also the bit-wise subset quantity **716** (two, in the present example). It should be noted that a target parallel hashing operation **739** is like the previously described parallel hashing operation **639** in that the invention might be practiced with variations of the specific steps therein which are presented here as features of the best presently known embodiment **10** of the present invention.

In a target memory setting operation **740** the reference hash table **644** is formatted such that each memory location **312** corresponding to a value of any of the target bit-wise subsets **736** is set. For example, if the first target bit-wise subset **736***a* were "000010" (decimal value 2) then the third memory location **312***c* in the reference hash table **644** would be set to "1", as is illustrated in FIG. **7**. As can be appreciated from the above discussion, the maximum number of memory locations **312** in the reference hash table **644** which can be set by this process is the quantity of target bit-wise subsets **736** (six, in the present example). However, since two or more of the target bit-wise subsets might coincidentally hash to the same value, a lesser quantity of memory locations **312** might also be set.

Now returning to a more detailed discussion of the bit-wise subset quantity determination operation **718**, the target memory **16** is to be configured to maximize the effectiveness of the filtering based on the quantity of multicast packets **210** of interest to the software of the host processor Therefore, the bit-wise subset quantity determination operation **718** attempts to determine (or, at least, to approximate) an optimal number of indices per packet (and, thus, the bit-wise subset quantity **716** discussed previously herein). The "optimal" number here means that which will minimize the number of "uninteresting" packets which match the set data bits **312** in the reference hash table **644** while matching all of the "interesting" packets **210**. In the best presently known embodiment **10** of the present invention, the following table is used to determine the bit-wise subset quantity **716**.

TABLE OF SUBSET QUANTITIES

| Addresses of Interest | Number of Hash Indices Bit-Wise Subset Quantity 716 |
|---|---|
| 1–2 | 5 |
| 3 | 4 |
| 4–9 | 3 |
| 10–16 | 2 |
| 17 or more | 1 |

The above table is offered here as a guide only, in that the "optimal" number of selected hash indices may vary in ways not presently contemplated. Furthermore, it should be noted that the above table is based upon an assumption that none of the target indices (the target bitwise subsets **736** in the best presently known embodiment **10** of the present invention hash to the same memory locations **312** in the reference hash table **644**. If, indeed, two or more of the target bit-wise subsets **736** did hash to the same memory location **312**, then additional hash indices could be added to increase efficiency without sacrificing speed or requiring additional memory or processing.

It should be noted that while the packet processing operation series **514** is accomplished in the hardware of the best presently known embodiment **10** of the present invention, the preliminary operation series (which can be accomplished at a more leisurely pace) is performed primarily by software of the host processor **20**. As can be appreciated in light of the above discussion, the preliminary operation

series will be repeated when the network **12** is reconfigured, when it is desired to communicate with additional members of the network **12**, or upon other occasions according to the needs of the user and the network **12**. The packet processing operation series **514** will be repeated whenever an incoming packet is detected from the network node **18**.

It should also be noted that, while the best presently known embodiment **10** of the present invention hashes each of the CRC values **628** and **728** to a common reference hash table **644**, the invention might be practiced with equal efficiency by hashing each of the CRC values **628** and **728** to its own individual hash table (not shown). Using the quantities of the example of FIGS. **6** and **7**, each of the individual hash tables would be 32 bits (memory locations **312**) large (one half of 64 bits, since it must be divided between the two target CRC values **728**). The individual bit-wise subsets **636** and **736** would then be 5 bits long (decimal value 0 through 31).

Various modifications may be made to the inventive improved packet filter **10** without altering its value or scope. For example, the quantity, size, and derivation of the plurality of bit-wise subsets **636** and **738** could readily be revised according to the parameters discussed herein.

All of the above are only some of the examples of available embodiments of the present invention. Those skilled in the art will readily observe that numerous other modifications and alterations may be made without departing from the spirit and scope of the invention. Accordingly, the above disclosure is not intended as limiting and the appended claims are to be interpreted as encompassing the entire scope of the invention.

## INDUSTRIAL APPLICABILITY

The improved partial packet filter **10** is adapted to be widely used in computer network communications. The predominant current usages are for the interconnection of computers and computer peripheral devices within networks and for the interconnection of several computer networks.

The improved partial packet filters **10** of the present invention may be utilized in any application wherein conventional computer interconnection devices are used. A significant area of improvement is in the inclusion of the parallel processing of a plurality of indices (bit-wise subsets **636**) of a packet

The efficiency of the filtering provided by the improved partial packet filter **10** is significantly improved, particularly for cases where the number of targets is small relative to the number of "buckets" (memory locations **312**). To compare the efficiency of the present inventive improved packet filtering process **510** embodied in the improved partial packet filter **10** with the prior art partial packet filtering process **410**, assume, for example, the following values:

Mb=64 (representing 64 memory locations **312** in the reference hash table **644**)

Cb=48 (representing a 48 bit candidate field **413** size—a typical size of the destination address **214**

Dn=4 (representing a bit-wise subset quantity **716** of four)

Then, the prior art partial packet filtering process **410** will partition the $2^{Cb}$ possibilities among 64 distinct buckets, one of which matches the bucket into which the single target falls. In the improved packet filtering process **510**, the four parallel hashing functions partition among 16 possible buckets each. The efficiency (Ef) for the prior art partial packet filtering process **410** would then be:

$$Ef=64/2^{48}=1/2^{42}$$

The efficiency (Ef4) for this example of the improved packet filtering process **510** is:

$$Ef4=64^4/(4^4*2^{48})=1/2^{32}$$

The efficiency Ef4 is better than the efficiency Ef by a factor of $2^{10}$ (1024), which is to say that only a thousandth as many (uninteresting) packets will be delivered to the next stage of filtering using the inventive improved partial packet filter **10** as compared to the prior art.

Filtering of packets may be accomplished through a combination of exact and partial match filters. Typically, one or more partial filterings will occur first, with the multiple dimensions of each filtering accomplished in parallel with each other (according to the present invention). Packets which pass through the inventive improved partial packet filter **10** may then be filtered using an exact match filter technique, such as "binary search lookup" of the filter data in a sorted table of acceptable filter data values. Furthermore, results of partial filtering can be used to determine which of many (possibly sorted) tables in which to search for the packet.

Accordingly, the inventive improved packet filtering process **510** may be applied more than once to each incoming packet **210** (in a first stage and a second stage). In such an example, configuration of the first stage partial filtering would involve specification of the number and type of hashing operations to be performed, along with the portion of the packet which is to comprise the filter data for each such operation, along with acceptable results for each. Multiple partial filterings may be configured with the specification including the logical relation to apply to the results of each filtering. For example, partial filtering A might be to apply the 32 bit CRC polynomial to the destination address field of an Ethernet packet, and retain the lowest order 3 bits—a value from 0 to 7. Partial filtering B might be to apply the 32 bit CRC polynomial to the source address field of the Ethernet packet, and retain the lowest order 3 bits. The logical relation might be to accept packets only for which the results of the first filtering (A) is either 2 or 4, and the result of the second filtering (B) is either a 3 or a 4. In a general case, one may expect the likelihood of arbitrarily filter data to "pass" the first filtering to be 2 in 8 (25%), since 2 of the 8 values from 0 to 7 are acceptable. Similarly, the likelihood of the second filtering "passing" such a filter is 2 in 8 (25%). Assuming that the two filterings are, as desired, truly independent, the likelihood of this arbitrary packet being accepted is the product of these, or 1 in 16. Note further that the specification of these "acceptable result sets" ({2,4} for A and {3,4} for B) requires 16 bits of information for full specification, where 8 bits indicate the acceptability/unacceptability of each of the 8 possible values of filtering A, and 8 additional bits indicate the acceptability/unacceptability of each of the 8 possible values of filtering B. Use of such multiple partial filterings may be especially effective in situations where filtering criteria are derived from independent portions of the filter data, such as filtering for all packets whose destination address OR whose source address is within a set of interesting addresses AND whose packet type indicates a particular protocol of interest.

Since the improved partial packet filters of the present invention may be readily constructed and are compatible with existing computer equipment it is expected that they will be acceptable in the industry as substitutes for conventional means and methods presently employed for partial packet filtering. For these and other reasons, it is expected

that the utility and industrial applicability of the invention will be both significant in scope and long-lasting in duration.

We claim:

1. A method for selectively forwarding a data packet and controlling the distribution of data packets in a computer network system, the data packet having a candidate field containing information about the data packet, the method comprising:

configuring a target memory of a controller to contain a target hash array in steps including;

aa determining a target field and extracting a plurality of target indices from said target field, the target indices being a binary number having a value;

ab setting memory locations in the target memory corresponding to the value of each of the target indices; and

processing the data packet in steps including:

ba extracting the candidate field from the data packet;

bb extracting from the candidate field a plurality of candidate field indices;

bc comparing the values of each of the candidate field indices to the target hash array; and

bd forwarding the packet when each of the values of each of the candidate field indices corresponds to a memory location of the target hash array which was set in step ab.

2. The method of claim 1, wherein:

step aa is accomplished in substeps including:

aa1 reducing the target fields to a plurality of target representative fields; and

aa2 selecting one or more target string subsets from the target representative field; and

step bb is accomplished in substeps including:

bb1 reducing the candidate field to a plurality of candidate representative fields; and

bb2 selecting one or more candidate string subsets from the target representative field.

3. The method of claim 1, wherein:

step ab is accomplished by causing only those memory locations in the target memory which correspond to the value of each of the target string subsets to contain a value of one.

4. The method of claim 2, wherein:

step aa1 is accomplished by applying a cyclic redundancy check algorithm to each of the target fields; and

step bb1 is accomplished by applying the same cyclic redundancy check algorithm to the candidate field.

5. The method of claim 2, wherein:

in step aa2 the target string subsets are selected by extracting a plurality of target bit-wise subsets from the target representative field; and

in step bb2 the candidate string subsets are selected by extracting a plurality of candidate bit-wise subsets from the representative candidate field.

6. The method of claim 2, and further including:

an additional process step preceding step ab wherein a subset quantity is determined, the subset quantity being the number of target string subsets to be extracted from each of the target representative fields and also the number of candidate string subsets to be extracted from each of the candidate representative fields.

7. The method of claim 6, wherein:

the additional process step is accomplished, at least initially, by selecting the subset quantity from a table of subset quantities.

8. The method of claim 2, wherein:

each of the target representative target and the candidate representative field are 32 bits in length.

9. The method of claim 1, wherein:

steps aa through ab are repeated when a change in the distribution of data packets is desired.

10. The method of claim 1, wherein:

steps ba through bd are repeated for each incoming data packet.

11. The method of claim 1, and further including:

an additional process step preceding step ab wherein a subset quantity is determined, the subset quantity being the number of target indices to be extracted from each of the target fields and also the number of candidate indices to be extracted from each of the candidate fields.

12. The method of claim 11, wherein:

the additional process step is accomplished, at least initially, by selecting the subset quantity from a table of subset quantities appropriate to a quantity of target quantities.

13. The method of claim 1, wherein:

the candidate field includes a target address field of the data packet.

14. The method of claim 1, wherein:

the data packet is a standardized Ethernet data packet.

15. The method of claim 1, wherein:

the target hash array is an unapportioned array such that each of the target indices is used to set memory locations in that unapportioned array.

16. The method of claim 1, wherein:

the target hash array is apportioned such that at least some of the target indices are directed to different portions of the target hash array.

17. The method of claim 1, wherein:

the target indices and the candidate indices are each a binary string of fixed bit length.

* * * * *